# AN EVALUATION SYSTEM FOR INTELLIGENT SMART BADGES

by

Yi Liu

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Canterbury

2006

Approved by _____
Chairperson of Supervisory Committee

_____

_____

_____

Program Authorized
to Offer Degree _____

Date _____

# UNIVERSITY OF CANTERBURY

## ABSTRACT

AN EVALUATION SYSTEM FOR
INTELLIGENT SMART BADGES

by Yi Liu

Senior Supervisor: Dr. Brent Martin
Department of Computer Science

Co-Supervisor: Dr. Mark Billinghurst

Human Interface Technology Laboratory New Zealand (HIT Lab NZ)

# Abstract

In this thesis we develop and test a software algorithm for an electronic smart badge system. The smart badge system we have developed has the ability to figure out the interests of people who wear the badge by using time and position information collected by the badge. The badge can also present feedback to the wearer, so that users may be guided to people will similar interests and so may have more effective conversations.

The smart badge system is based on an inference system which uses a Bayesian network. Evaluation of the system was challenging because there were no completed badges that could be used. To overcome this, we developed a simulation of crowd behaviour in a conference setting. We tuned the parameters of the model using several test situations and the final simulated behavior appeared realistic.

Compared to other smart badge systems, our work is unique because it is able to enhance conversation by the real time inference of common ideas or interests of the conversion participants.

# ACKNOWLEDGMENTS

My most sincere gratitude goes to my supervisor Senior Lecturer Brent Martin, and associate supervisor Associate Professor Mark Billinghurst, for their guidance, support and kindness. Without their contributions I would not be able to conduct this research.

I would also like to thank all friends who support me. In particular, I would like to thank Malcolm Shore and his wife Sarah for their continuous support throughout my academic career.

Special thanks to my family members. Especially, I would like to thank my girlfriend Zhiqi Tu and my grandmother. But most of all I am eternally grateful to my parents for risking and sacrificing all to give me a brighter future.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

Chapter 1

# INTRODUCTION

In our modern world, people widely use technology to support remote communication. Despite this, most social interaction still happens when people meet face to face. We can meet someone unexpectedly, for example, in a hallway or a lift. We may meet people we do not know or with whom we are not familiar. In fact, we do not know the majority of people we encounter. We can also meet people with whom we are familiar. “Any encounter with friends, family and strangers is a chance for striking up a conversation and for exchanging information” [1].

When people attend conferences they usually wear a name badge to tell other people who they are and where they are from. The printed name badge is an easy way of sharing this information in a public setting. With modern technology it is possible to create an electronic version of the paper name badge with extra functionality. Recently people have begun developing electronic name badges with some additional functions, often called Smart Badges. The Smart Badge is a small-scale smart device that can be worn by conference attendees or people in other public places.

There have been a number of different research groups that have developed variations of the electronic name badges. Some use on-board Infrared (IR) and radio frequency (RF) hardware to provide several levels of security for communication among the wearers, and also track the behaviour of the wearer.

The goal of our Smart Badge system is to create an electronic conference badge. This badge can be used to seamlessly collect information, like business cards, in real-time. It achieves this purpose by using an infrared sensor to exchange the ID numbers between a pair of Smart Badges facing each other; the time spent in contact is also recorded. This data is periodically uploaded to a central server via an RF link where the IDs can be matched to contact details or other predefined data. The data can then

be sent out post-event and sorted by contact time to give a list of people talked to in a likely order of importance [1].

Smart Badges could be used in other interesting ways. In our work we are interested in developing an ‘‘Attendees’ Memes Inference”, which can infer the opinions and interests of the wearer in real time. A Meme is a succinct idea or opinion [2]. The Smart Badge can then assist the wearer in finding the other attendees who have the same interests as the wearer, similar to a ”Friends of Friends“ function described in [2]. What’s more, the intelligent system could group the attendees according to their memes so that those attendees can make friends more easily.

Figure 1.1 shows the smart badge system. The system has a server computer and a number of electronic smart badges. The badges not only collect data from the objects in the environment, but they also are able to communicate with each other by infrared. Furthermore, the badges can also exchange data with a central server computer server via RF link.

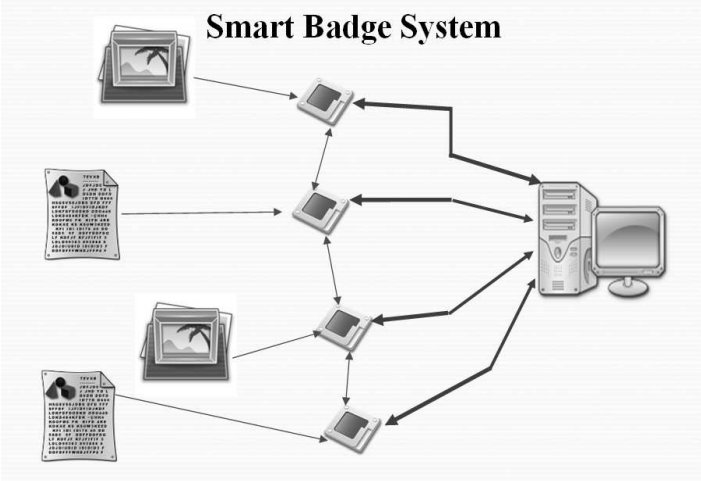



Figure 1. 1: Smart Badge System

## 1.1 Implementation of an Intelligent Smart Badge System

To implement a Smart Badge system there are four parts that must be developed:

- The hardware device
- Central server software
- A simulation application
- The data communication system

The hardware device supplies the physical access to the user. Since a small electronic badge is not able to provide much processing power, we need to have a central server system which can perform matchmaking services from the data sent from the badges. The simulation application will be used as an evaluation tool. We will present a virtual conference to verify the badge behaviours. In this research, we combine the central server application and the simulation application in an evaluation system. Finally, the data communication system provides a secure network connection so that others who are not attendees of the conference cannot access our system using other electronic tools.

### *1.1.1 The Hardware*

The implementation of the hardware for the Smart Badge was performed by another HIT Lab NZ[1] student. The research areas covered in his work included the electronics of the hardware design and also the creation of a new Medium Access Control (MAC) protocol for the RF network to maximise the battery life of the Smart Badges.

The HIT Lab NZ Smart Badge (Figure 1.2) has an LCD screen to display the wearer's name to others, and any other data an application may require. It also has four bi-colour LEDs, two buttons and a buzzer for use with future applications.

[1] HIT Lab NZ: Human Interface Technology Laboratory New Zealand.

While electronic business card exchange between badges was the initial application many others have been suggested, including using the LEDs on the badge to show a match of interests between Smart Badge users, embedding Smart Badges in company booths to collect data on how long potential customers visited the booth, and augmenting social networking by giving the names of other people with similar interests to those already met [1].

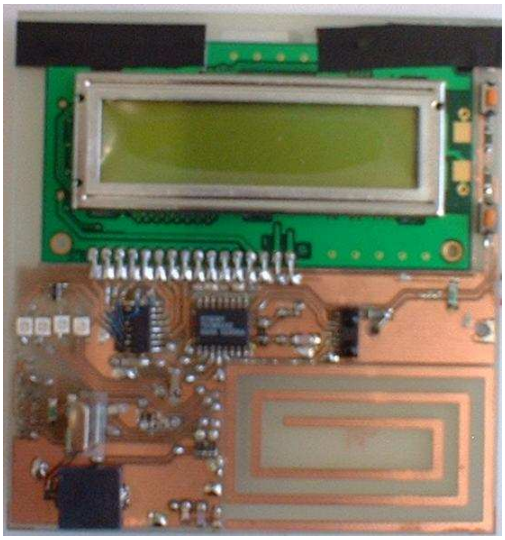


Figure 1.2: the HIT Lab NZ Smart Badge hardware

### *1.1.2 The Central Server*

The central server software is the main contribution of this thesis. We call the server application the intelligent system. The intelligent system has two sub-applications which are the inference module and the matchmaker module.

The goal of the inference module is to infer the conference attendee's interests through the data collected by his or her badge. For example, when the attendee stops in front of a poster, and watches it, the badge's infrared sensor can detect the signal from the sensor fixed on the poster. Therefore, the badge can track how long the attendee spends at the poster and what keywords the poster contains. Consequently, the data will be sent to the server, and updated by the inference model. A numeric level will be calculated which represents how much the attendee is interested in the

poster keywords. The input data of the inference module could be the time the attendee spent on a certain poster, and what keywords the poster contains.

The goal of the matchmaker module is to use a description-logic matchmaking algorithm to calculate the similarity in interests of two attendees. That is to say, when two attendees encounter each other, the infrared sensors of their badges can detect the signal before they meet. Therefore, the badges can send the IDs of two attendees to the server. The feedback containing the similarity will be sent back to the attendees, and the badges may display the keywords they have in common. This way the attendees can find out about each other before they start to talk. The input data of the matchmaker module are the attendee's interests as generated from the inference module.

### *1.1.3 The Simulation Application*

There were not enough real Smart Badges in the HIT Lab NZ to be used for large scale testing so simulation software is used to evaluate the intelligent system. It is also much easier to change the software to simulate adding new sensors etc, than modifying the badge hardware.

The simulation application simulates a real conference environment. In the simulation, the layout of the conference meeting-rooms is shown on a desktop screen along with icons representing the attendees. There can be hundreds of agents moving around in the meeting-rooms, walking along a path, all wearing simulated Smart Badges. The code also simulates the data communication and exchange among badges. Networking security will also be presented in the simulation application.

### *1.1.4 Data Communication*

A special data communication protocol will be built for our project that can provide steady, fast and secure data communication. There are various communication methods based on the different relationships between the attendees. These communication methods can be used to provide a secure network in our project. For example, if the intelligent system has inferred that the relationship of two attendees is

‘could be a good friend’, then the communication system just uses the low security level to transfer the data. They can exchange personal information such as their email address or the telephone of their office. Consequently, the protocol will be a brand-new protocol for Smart Badges, providing highly secure protection of the personal data. The algorithm for the protocol will be created by another student from the computer science department.

## 1.2 Objective

Before describing the main research objectives, the functions of the badge have to be mentioned. Currently, we have designed three functions, which are ‘Attendees’ Memes Inference’, ‘Seeking Friends’, and ‘Social Relationship Inference’. The interpretation is as follows:

- ‘Attendees’ Memes Inference’

The function can analyze the wearer’s interests by using a Bayesian Networking algorithm, rather than the wearer programming their interests by themselves.

- ‘Seeking Friends’

This is an introduction service which uses matchmaking algorithms to find people with similarities in their profiles.

- ‘Social Relationship Inference’

This function infers social relationships between people based on proximity and interaction behaviours. Being able to infer relationships in this way will augment a variety of existing services that currently require users to manually quantify existing relationships. When two people encounter each other, they cannot only know each other names (on the badge), but they can also know what the relationship between them is. By inferring the social relationships, the intelligent system can group the attendees according to their purposes for attending the conference and the memes they have.

To achieve these functions, five research objectives need to be taken into account. They include the aspects of hardware, simulation, the core application, data communication and evaluation of the effectiveness of the Smart Badges. Hardware design and data communication are not the main research focus of this thesis. In our work we have designed and developed a simulation system and the central server in order to test and evaluate the Smart Badge.

### *1.2.1 Smart Badge Hardware Design*

The design of the Smart Badge software and applications which run on the badge is dependent on the basic properties of the hardware, such as memory, the process speed of CPU, the function of CPU, and so on. Therefore, it is crucial to design suitable hardware for the project. As mentioned previously, the hardware design is out of the scope of this thesis and is being completed by another HIT Lab NZ student.

### *1.2.2 Developing a Simulation Application*

The simulation application simulates a real conference environment. The simulated conference space has 5 meeting-rooms. In each meeting-room, there are 4 posters on a similar topic. There will be about 50 agents moving around in the meeting-rooms, all wearing simulated Smart Badges. The agents with badges can look at a poster, and the application records the time spent looking at posters for every agent. Moreover, the agents are allowed to talk to each other and the application also records the ID of who they are talking to. The agents and their behaviours will be shown on the desktop screen. The code also simulates the data communication and exchange among badges. Networking security will also be presented in the simulation application. The simulation application is developed entirely as part of this thesis.

### *1.2.3 Developing an Intelligent System*

The intelligent system is the core application of the project, and it is able to infer the attendees' interests and figure out the common memes in real time. Four functions of the system have been proposed, although in the future, the intelligent system would probably have several other functions. Furthermore, a database is a vital element of

the intelligent system. The system will generate a database for all badges in order to record the agent profile, the similarity of interests of each agent with the others, and the weight of the memes. The intelligent system is also developed entirely as part of this thesis.

### *1.2.4 Building Security Protocols for the Data Communication*

In this system private information must be exchanged in a secure way. Therefore, this project will also use related research for implementing a security encryption/decryption protocol for real-time detection and exchange of private information in a limited knowledge or a focused interaction. The security protocol will be created by another student.

### *1.2.5 Evaluating the Effectiveness of the Intelligent System Using the Simulation Application*

After the development of the central server and the simulation application, the effectiveness of the intelligent system needs to be evaluated. We will design an evaluation system that includes a simulation application to test the central server. The simulation is to simulate the real environment which the smart badges are going to be used in.

## 1.3 Methodology

The design of hardware and the network security protocol is not developed as part of this thesis. Therefore, we will not mention these two topics in this section. In this section, we illustrate the methodology and the thesis structure as well.

### *1.3.1 Simulation Application*

To develop the simulation application it is necessary to design a reasonable structure and to choose a suitable development tool and language. The MFC (Microsoft Foundation Class) C++ library can be used to provide the application with a standard

Microsoft windows style, and a flexible framework. OpenGL is a well-known graphics library which can be used to present the rotation, transformation and movement of agents. Consequently, OpenGL will be the graphic development tool used in our project. Detailed information will be illustrated in Section 3.4.

The purpose of the conference simulation is not to show the detail of agents and other objects, but to simulate the agents' behaviour. Accordingly, it is not necessary to use a full 3-D environment in order to make a movement pattern for each agent. As a result, we choose a 2-D view for the presentation of the simulation. The procedure for designing the simulation is shown in Section 3.2. Chapter 3 also covers the motivation and functionalities of the simulation (Section 3.1 and Section 3.3).

### *1.3.2 The Intelligent System*

The intelligent system is not only the most vital, but also the most complex and difficult part of the project. Firstly, in Section 4.1 we analyze what the badge and the system should do respectively. The badge uses its sensors to collect where and how long the wearer stops, and how long and who the wearer talks to. The system architecture is described in Section 4.2.

The profile of each wearer is very important for achieving the main badge functions, because all calculations are based on this profile. For example, the calculation of the similarity in interests of two users talking to each other is based on the profile values. In this project, the profile includes the user interests, and other basic information. We will group these interests into five groups and create an interest hierarchy. Section 4.3 will talk about these points. Finally, Section 4.4 describes the procedure for implementing the Bayesian network which provides probabilistic inference. A Bayesian network is used to calculate the probability of the evidence we are interested in, and update the probabilities of other evidences in the system.

### *1.3.3 Testing and Evaluation*

Testing is a crucial step in the development of a software application. The plan for testing and evaluation is shown in Chapter 5. This chapter explains a concept we call

artificial data which is used to provide the testing data set to the Bayesian network. Chapter 6 shows the results of testing and evaluation.

Finally, we describe background knowledge and related work in Chapter 2. The background knowledge includes an introduction of smart badge techniques and the definition of Bayesian networks. We illustrate several similar electronic badges which have been developed by other researchers. Next, chapter 3 outlines the implementation of the simulation application. The development of the central server is discussed in chapter 4, while chapters 5 and 6 are about the experiment and chapter 7 contains the conclusion.

## 1.4 Research Contribution

Although there are many research projects in Smart Badge technologies there are a number of differences between our research and earlier projects. First, our badge and intelligent system can infer the opinion and interest of attendees in real-time while the badges are moving around, rather than using the personal information that is programmed before the conference beginning. Previous Smart Badge technology has not been developed and evaluated that includes both matchmaking and inference. Another difference is that our Smart Badge is able to enhance the conversation using the results of this inference. The result of the inference will be used as the input into the matchmaker module so that we can figure out the similarity of wearers. Through providing feedback that includes common ideas or common interests to wearers, the conversation will be enhanced.

Another contribution of this project is to provide an example of how Bayesian networks can support Smart Badge applications and how a simulation system can be used to evaluate the application. We design a particular Bayesian network to infer the interests of the attendees. The previous attendees' behaviours are taken into consideration as an evidence of Bayesian network in order to the server can get the correct result. We also use a simulation application instead of a difficult and expensive deployment of large amounts of hardware.

In this project we simulate a potential conference application. The Smart Badge could be used in the many different public situations, such as a conference, an exhibition or a museum. Thus this research is the first step in creating a brand-new product which can be applied in conference or museum in order to provide an intelligent guide to attendees.

Chapter 2

# RELATED WORKS AND BACKGROUND CHAPTER

This chapter will describe the related work of other researchers, and the background material for the Smart Badge project. The related works cover some similar electronic badges, from which we obtained the basic ideas. The background knowledge includes crowd simulation which is the main technique used in the simulation application, the description logic matching algorithm which performs the agent matchmaking, and Bayesian networks which provide the inference techniques for the Smart Badge system.

Section 2.1 covers related works in the areas of electronic badges and past inference applications. Section 2.2 introduces related wearable computing techniques used to augment human interactions and enhance human communication. In Section 2.3, we will discuss crowd simulation, and in particular the use of intelligent agents. Matchmaking techniques will be discussed in Section 2.4. Section 2.5 will review three main machine learning methods and compare those to the Bayesian network which is our approach. Section 2.6 will describe in detail the definition of a Bayesian network, and four situations of learning Bayesian networks from data. Section 2.7 mainly discusses Bayesian network inference algorithms, and a comparison of them. Finally, section 2.8 provides a summary and concludes this chapter.

## 2.1 Related Work with Electronic Badges

There have been a number of previous research projects whose objective is to develop electronic badge technologies. Some of the resulting technologies have been commercialized. The ones most relevant for our project are: nTag [5], Thinking Tag [2], Meme Tag [2] and the wearable remembrance agent. We review these technologies and also two previous applications about inference.

### *2.1.1 Related Tags*

#### Meme Tags

Memes are an idea or opinion expressed as a short piece of text [2]. Meme Tags are smart badges developed at the MIT Media Lab which can be used to help conference attendees build a better shared understanding of the whole conference community (Figure 2.1) Identifying the communities of interest at a conference can create a stronger sense of group identity.

Before the conference starts, a significant number of memes need to be loaded into the Meme Tags. Then when people wear Meme Tags and move around the room, memes that two conversing people's tags have not seen before would be presented in their respective tags. They can use buttons on the badge to accept this new meme if they like it, or reject it. Each time attendee meet, their tags create a record to indicate whom they met, and which memes were exchanged or rejected. In addition, during their conversation, the Meme Tags also exchange records of all other conversations stored in the tags. Thus, each Meme Tag collects a sample of the conversation records from throughout the entire community [2].

When attendees go to a kiosk, all records in their tags are dumped into a Meme Server Database. Although not every attendee would visit the kiosk, each attendee's tag contains a representative sample of conversation records from the entire group [2]. A substantial portion of conversation records could be collected.

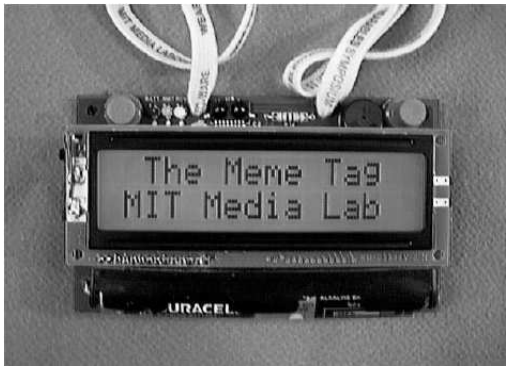



Figure 2.1: the Meme tag

## nTags

The nTag is an interactive name tag, developed by the company nTag Interactive of New York [5]. It is a PDA-like device equipped with a black-and-white LCD display, infrared port, and several buttons that the users can choose their operations, see Figure 2.2.


Figure 2.2: the nTag

When you come to an nTag event, a fair amount of information about who you are, what your interests are, or what you want to know/accomplish in this event will be preloaded into your badge. Whenever two people talk their badges communicate via infrared to quickly decide what they have in common, and display common information in the screen. In addition, the nTag also records whom you talk to, and how long your conversation lasts.

The nTag makes it easy for people to swap contact information. If you want, you can use the button to choose with whom you want to exchange your information. And other people can do the same thing to you as well. Thus, after this event, attendees will receive email with the information that they collected already in an electronic form [5]

#### Thinking Tags

The Thinking Tag [2] is an electronic badge which is also able to dispense information at a time when the tag is useful and relevant. When two persons wearing the thinking tags meet each other face to face, the tags can present how much they have in common.

The Meme Tag, nTag and Thinking Tags have proved to be quite useful and helpful in conferences, however currently there is no difference in the information sent between your good friend and a stranger in the same conference, and the users' interests must be explicitly programmed into the badge. In this thesis work, we will build an intelligent system which automatically infers the attendee's interests or opinions, rather than explicitly programming the interests and opinion data into a database.

### *2.1.2 Related Inference Applications*

#### The Museum Wearable

Wearable computers provide enhanced functionality over what is possible with the basic Smart Badge system. An example of this is the Museum Wearable [6] which is a wearable computer that orchestrates museum audiovisual narration as a function of the visitor's interests. Visitor interests are gathered from his or her physical path in the museum and the length of stops in front of museum artefacts. The wearable is made from a lightweight small computer that people carry inside a shoulder pack. (See Figure 2.3). It provides an audiovisual augmentation of the surrounding environment using a small eye-piece display attached to conventional headphones. Using custom built infrared location sensors distributed in the museum space, and statistical mathematical modeling, the museum wearable builds a progressively refined user model and uses it

to deliver a personalized audiovisual narration to the visitor. “This device will enrich and personalize the museum visit as a visual and auditory storyteller that is able to adapt its story to the audience’s interests and guide the public through the exhibits” [6].

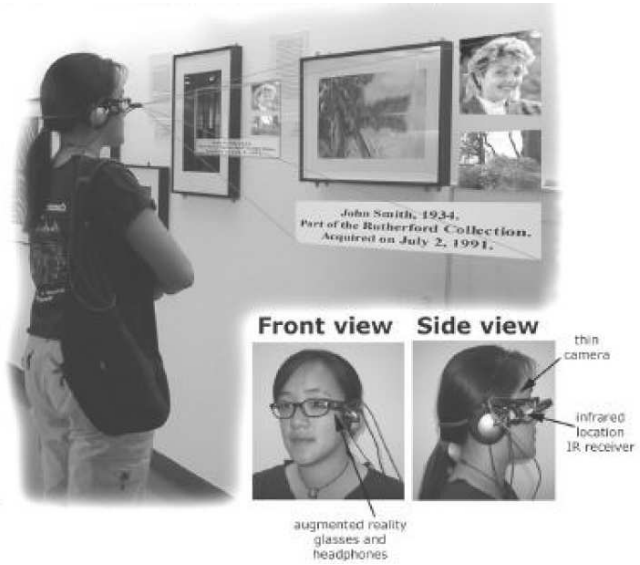



Figure 2.3: the shoulder pack for the museum wearable device

The strength of the wearable device is that it can provide an augmented object to the visitor, since the device can infer the interests of the visitor in real time according to his physical path. However, it only works in smart environments with location sensors and does not provide support for enhanced communication by taking the interaction between people into consideration.

Although Smart Badges cannot augment the objects the wearer is interested in, they can augment the communication between people due to Smart Badges being able to infer the similarity of interests of people.

## 2.2 Crowd Simulation

We are interested in developing smart badges for exchanging information among people in a meeting situation. Before we can develop a fully functional real system, we would like to simulate our system to evaluate the communications protocol and server

software. In order to do this we need to simulate a conference meeting. In our simulation, there are about 50 agents in a virtual conference environment which has 5 virtual rooms. Therefore, some virtual rooms become crowded, and the agent motion is constrained. To model this we will use Crowd simulation technology.

“Crowd simulations are becoming increasingly important in the entertainment industry. In movies, they can be used to simulate the presence of real humans. For example, in the movie Titanic, extensive use was made of virtual people, whose movements were generated from a library of pre-captured motions.” [4] Moreover, they have been used to train the military and policeman. Crowd motion simulations also have been utilized to support architectural design both for everyday use and for emergency evacuation conditions. Simulations have been used to investigate physical aspects of crowd dynamics [7]. Finally, simulations can be used in the sociological and behavioural contexts.

In this section, we will focus on the intelligent agents used for the simulation of the human crowd’s dynamic behaviours. Firstly, we discuss what the crowd simulation is. Secondly, we present intelligent agents. Finally, three basic behaviours are discussed.

### *2.2.1 What is The Crowd Simulation?*

Defining crowd simulation is difficult because there is no standard definition. Firstly, let us see the density of ‘crowd’ (Figure 2.4 [8]).

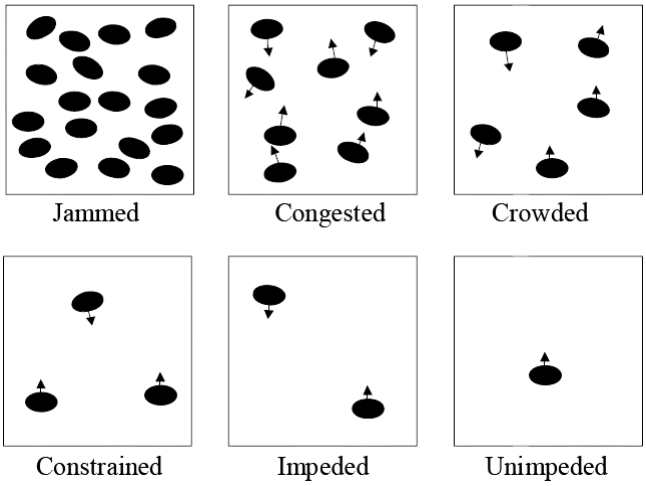



Figure 2.4: the density of 'crowd'

As we can see, the agents cannot easily move and rotate in a *jammed* environment. The second level is *congested*, in which it is impossible for agents to move with high speed. The third one is *crowded.* The agent has more space than the *congested* one. Constrained, Impeded and Unimpeded all belong to crowds which are of low density. The Agents in these latter three situations are not too constrained, unlike those in the first three situations.

We cannot provide a fixed definition about crowd simulation because different applications have different descriptions. However, they have several common points as follows:

- A Massive Crowd Simulation is a computer simulation of a crowd of agents.
- Agents are digital characters with a certain artificial intelligence.
- The individual agents can act on their own, they don't need to be pre-programmed or scripted, and they respond without external control [10].

Agents involved in the simulation are digital characters with some artificial intelligent behaviour. In the next section we describe this in more detail.

### *2.2.2 Intelligent Agents*

An agent can be viewed as perceiving its environment through sensors and acting on that environment through effectors. A human agent has eyes, ears, and other organs for sensors, and hands legs, mouth, and other body parts for effectors. A generic agent is shown in , taken from [11].

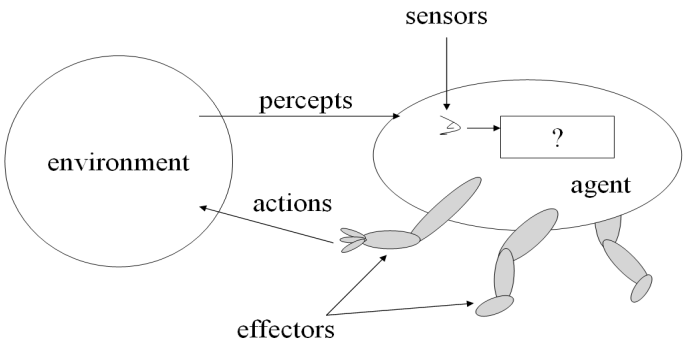



Figure 2.5: agents interact with the environment through sensors and effectors

A rational agent should do the right thing in the right place at the right time. The result will cause the agent to be successful. Hence, this leads to the following definition of an ideal rational agent: “For each possible percept sequence, an ideal rational agent should do whatever action is expected to maximize its performance measure, on the basis of the evidence provided by the percept sequence and whatever built-in knowledge the agent has” [12].

When the agent perceives the environment using its sensors, the artificial intelligent behaviour may be generated by a combinative analyse of states, rules and other conditions as well. The behaviour of the agent is determined by many aspects, especially the environment the agent is in. Now we discuss the three basic artificial intelligent behaviours.

### *2.2.3 Behaviours*

#### Motion Behaviour

The term *behaviour* has many meanings. It can mean the complex action of a human or other animal based on volition or instinct. It can also mean the largely predictable actions of a simple mechanical system, or the complex action of a chaotic system [3]. The behaviour of an autonomous agent (AA) can be divided into several layers. Figure 2.6 [3] shows a division of motion behaviour for AA agent into a hierarchy of three layers: *action selection*, *steering*, and *locomotion*.

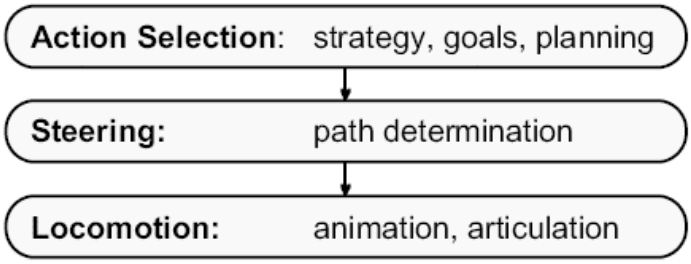



Figure 2.6: a hierarchy of motion behaviours

*Action Selection*: noticing that the state of the world has changed and setting a goal.
*Steering:* see the following part.
*Locomotion*: It converts control signals from the *steering* layer into motion of the agent's "body". This motion is subject to constraints imposed by the body's physically-based model.

#### Collision Avoidance

Collision Avoidance is a huge research area in crowd simulation and it governs an agent's interaction with other objects in the environment [12]. For example, in a traffic simulator, an agent, such as a vehicle, must have the capability to avoid objects such as walls or other vehicles. It should have the capability to determine how to avoid accidents. In this case, the agent is autonomous to make all decisions, rather than programmed by the designer.

Successful collision avoidance has several steps. Firstly, the agent is able to predict where the point of potential collision is. For the static objects, this is easy to figure out the point. On the other hand, for the dynamic object the agent has to figure out the point they will encounter each other. The agent calculates and obtains the result by the direction and the speed of the dynamic object. Secondly, after the agent detects there is a potential collision, it will determine the rotating angle and speed so that it can safely avoid the collision. Finally, the agent calls the steering function to implement the avoidance.

### Steering

In the simulation, the agents need to follow a particular path. When they encounter an obstacle or each other, they need to perform a steering action and slow down so that they can avoid the obstacle or wait for other agents give way. After avoiding the obstacle, they need to speed up. Moreover, the agents should not be able to walk through virtual walls.

*Steering* can be classified into six categories as follows:

1. *Path Following Steering*: “It is to traverse the path in a given direction (entering on the left, exiting on the right) while keeping its centre in the appointed region” [13].
2. *Queuing Steering*: “it is to produces *braking* (deceleration) when the vehicle detects other vehicles which are: nearby, in front of, and moving slower than itself” [14].
3. *Seek and Flee Steering:* “*Seek* attempts to steer an agent so that it moves toward the goal. *Flee* attempts to steer an agent so that it moves away from the goal” [15].
4. *Wall Following Steering:* “it is to remain a given distance from the "wall" as it moves” [16].
5. *Wander Steering:* “Wandering is a type of *random steering* which has some long term order: the steering direction on one frame is related to the steering direction on the next frame” [17].
6. *Obstacle Avoidance Steering*: “it is to keep a distance from the point of potential collision” [18].

## 2.3 Matchmaking

Matchmaking attempts to determine whether two objects refer to the same features, profile or entities. Matchmaking has become a very important tool in many aspects such as scientific research and industry. For example, there is a matchmaking software agent in Business-to-Business (B2B) e-commerce systems [19]. Furthermore, the object matching system [20] can play an important role in the information management, including information integration, data warehousing, information extraction, and text joins in databases.

It is impossible to discuss all matchmaking techniques here. We present a matchmaking technique using a user profile, particularly profile comparison and description logic. This technique can evaluate the similarity of two objects through comparing their profiles. This section presents the matchmaking which can test the similarity of our virtual agents. Firstly, we will introduce the description logic. After that, we will briefly present the basic matchmaking technique.

### *2.3.1 Introduction to Description Logics*

“Data and knowledge are based on a model of part of the natural world. For example, some models are built from the individual object which is related by the relationship and grouped into certain classes capturing the commonalities among their instances” [21]. *Description Logics (DLs),* also called *terminological logics,* are a kind of language which is used to build and access the preceding described models. DLs are a family of logic-based knowledge representation (KR) formalisms designed to represent and reason about conceptual knowledge [22]. Its main feature is that the *concept* can be defined *intentionally* in terms of descriptions that specify the characters that objects must satisfy to belong to the concept. DLs have found applications in many areas such as modeling database schemas and the semantic web [21].

Elementary descriptions are *atomic concepts* and *atomic roles*. Complex descriptions can be built from them inductively with *concept constructors* [23]. Attributive Language (AL) is used to present the descriptions.

Concept descriptions in AL are defined by the syntax rules [23]. For example, let letters A and B represent atomic concepts, letters C and D concept descriptions, and letter R as atomic roles. See Figure 2.7 [23]:

$$
\begin{array}{lll}
C, D & \longrightarrow & A \mid \qquad \text{(atomic concept)} \\
 & & \top \mid \qquad \text{(universal concept)} \\
 & & \bot \mid \qquad \text{(bottom concept)} \\
 & & \neg A \mid \qquad \text{(atomic negation)} \\
 & & C \sqcap D \mid \qquad \text{(intersection)} \\
 & & \forall R.C \mid \qquad \text{(value restriction)} \\
 & & \exists R.\top \qquad \text{(limited existential quantification)}
\end{array}
$$

Figure 2.7: the syntax rules

For example, to describe a person, we could create two atomic concepts $Person$ and $Female$. We could describe a man using $Person \cap \neg Female$, a woman using $Person \cap Female$. We suppose $\exists hasJob$ is an atomic role, and $Full$ is another atomic concept. We describe a man with a full time job using $Person \cap \neg Female \cap \exists hasJob.Full$.

### *2.3.2 Profile Matchmaking*

Many applications require profile matchmaking, such as job recruitment or dating systems. The purpose of profile matchmaking is not to find an exact match of profiles, in fact such an exact match may be impossible to find. The purpose of profile matchmaking is to find the best possible match [26]. Such a non-exact match has to take missing information and conflicting information into consideration. Furthermore, if there are several matches that are possible, the matchmaker has to list them in most-promising order, in order to enhance the probability to get a successful match.

## 2.4 An Overview of Three Machine Learning Methods

“Machine learning is a sort of computer programme which is able to learn from the prior experience (E) according to some tasks (T) and performance measure (P), if and

only if its performance at T as measure by P improves with E". [24] Inductive learning and deductive learning are two main methods of machine learning.

Inductive learning tries to estimate or create an evaluation function from a set of examples. Deductive learning does not need additional input, but improves the agent's performance over time. Another learning method called connectionist learning represents the data structure as a set of nodes which are connected through the weighted links.

In this section, we discuss three machine learning methods; decision trees, Instance-Based Learning (IBL) and neural networks. Those three methods respectively belong to inductive learning and connectionist learning. At the end of this section we compare the Bayesian network to those three methods, and discuss why we choose Bayesian networks as our approach.

### *2.4.1 Decision Tree*

A decision tree is a simple structure of inductive learning. When an instance of an object or situation is given, the return value of the decision tree is "yes" or "no" or other multi-scale values. Therefore, decision trees provide a classifier function. The branch nodes of the tree represent the tests or some aspects of the instance, and they are also classifiers. Decision trees try to learn the goal predicate which is a set of implication sentences.

The disadvantage of a decision tree is that:

- If there is no extent example in a branch, then a default preset will be applied.
- Some other methods have to be applied to determine what classification should be used, when noise occurs.

### *2.4.2 Instance Based Learning (IBL)*

IBL is also a kind of inductive learning and is a machine learning method. IBL performs classification through matching the new instance with the case added to memory. It determines which case is similar to the new instance.

The disadvantage of the IBL is:

- Slow classification time. If the number of cases in memory is large, then the classification time increases.
- Noise in the data can affect the classification result because IBL may be confounded by noise.
- IBL assumes all attributes have same importance.

### *2.4.3 Neural Networks*

A neural network can consist of a set of nodes and weighted links, in which the input of some nodes comes from the links, and the input of others may come from the environment directly. The output of the network is generated by some nodes. Learning is achieved by adjusting the weights on the links.

A neural network can be seen as a combination of a set of units. Each unit has an activation level which is a set of weighted inputs, and an activation function which is to calculate its activation level at the next time step. The activation function figures out the weighted sum of the node's inputs. The weighted sum is a strictly linear sum, but the activation function may be not linear. If the activation function's return value is larger than a threshold, the node fires.

### *2.4.4 The Reason Why We Choose Bayesian Networks*

In this subsection, we discuss the weakness of decision tree and IBL, and compare Bayesian networks to neural networks, in order to show that a Bayesian network is a suitable solution for our research.

#### The Weakness of Decision Tree and IBL

Decision trees and IBL have strengths and weakness. Firstly, both cannot perform a better performance across the complete spectrum of problem domains compared to Bayesian network [26]. In our research, the attributes are not identically important. For example, the value of the attribute called Interest Group affects the final result more

than the attribute called Prior Keyword. In the meantime, its effect is less than the attribute called Duration. Finally, although IBL spends a short time on learning, its classification time increases when more examples are added to memory, because it has to match the new instance with every example in the memory.

Therefore, we think Bayesian networks are better than decision trees and IBL is the ideal learning method for our research.

### A Comparison of Bayesian Network and Neural Network

Bayesian networks and neural networks are both types of connectionist learning. We compare them according to representation, inference system and learning system.

They both are attribute based representations. They both can generate discrete or continuous output. However, a Bayesian network uses localized representation, while a neural network is distributed. In a Bayesian network, the nodes represent propositions with clearly defined semantics and relationships with other nodes. On the other hand, in a neural network, the nodes do not represent propositions, while the computation does not deal with them in semantically meaningful way.

A Bayesian network can apply two kinds of activation, the value a proposition can take and the probabilities assigned to each value. Although a neural network's output can be a value or a probability, it cannot do both at the same time.

As for learning, Bayesian networks may be better than neural networks, because it is easier to give them prior knowledge. Moreover, a Bayesian network is a localized representation. It is easier to converge because it is just affected by a small number of other propositions.

Furthermore, designing a Bayesian network's topology for a given problem is easier than a neural network.

In short, a Bayesian network represents the data structure using a directed acyclic graph which is able to efficiently reflect the causal relationship among variables.

Meanwhile, there is causal relationship between the variables applied in our application. For example, how much you like an object will determine how long you are willing to spend on the object. Moreover, from the personal view, I like to use a graph model to solve a real problem because I think the graph structure can represent the logic and relationship of the problem more clearly. Thus, we determine a Bayesian Network is the best choice by the comparison of Bayesian Networks, decision trees, IBL and neural networks.

## 2.5 Bayesian Network

In this section, we will discuss the definition of Bayesian network as well four situations to learn Bayesian network from data definition of Bayesian Network

**Bayesian Rule:**

$$P(\theta \mid D, \mathrm{H}) = \frac{P(D \mid \theta, \mathrm{H})P(\theta \mid \mathrm{H})}{P(D \mid \mathrm{H})}$$

Here, $\theta$ denotes the unknown parameters. *D* denotes the data, and H denotes the overall hypothesis space. We call the marginal probability $P(\theta \mid \mathrm{H})$ the prior probability. We call $P(D \mid \theta, \mathrm{H})$ the likelihood of $\theta$. The conditional probability $P(\theta \mid D, \mathrm{H})$ is called the posterior probability. Finally, $P(D \mid \mathrm{H})$ is the evidence or the marginal likelihood. Therefore, we can rewrite the rule again as followed:

$$posteriror = \frac{likelihood \times prior}{evidence}$$

**Definition of Bayesian Network:**

Casual relations also have a quantitative value associated with links, which we call a link's 'strength'. A is a parent of B. If we use a probabilistic approach, we can let $P(B \mid A)$ be the strength of the link. If C is also a parent of B, we can obtain two probabilities, $P(B \mid A)$ and $P(B \mid C)$.

A Bayesian network [26] for a set of variables $X = \{X_1, X_2, ....., X_n\}$ consists of

(1) A network structure $S$ that encodes a set of conditional independence assertions about variables in $X$.

(2) A set P of local probability distributions associated with each variable.

Together, these components define the joint probability distribution for $X$ . The network structure $S$ is a directed acyclic graph. The nodes in $S$ are in one-to-one correspondence with the variables $X$ . We use $X_i$ to denote the variable and its corresponding node, and $Pa_i$ to denote the parents of node $X_i$ in $S$ as well as the variables corresponding to those parents. The lack of possible arcs in $S$ encodes conditional independencies. In particular, given structure $S$ , the joint probability distribution for $X$ is given by

$$P(x) = \prod_{i=1}^{n} P(x_i \mid Pa_i)$$

The local probability distributions $P$ are the distributions corresponding to the terms in the product of the preceding equation. Consequently, the pair $(S, P)$ encodes the joint distribution $P(x)$ .

Bayesian network cannot only use diagnostic reasoning, but it also can make inferences. We will discuss inference in Bayesian network in the next section. The tasks a Bayesian network can perform are listed as follows [11]:

- Making decisions based on probabilities in the network and on the agent's utilities.
- Deciding which additional evidence variables should be observed in order to gain useful information.
- Performing sensitivity analysis to understand which aspects of the model have the greatest impact on the probabilities of the query variables (and therefore must be accurate).
- Explaining the results of probabilistic inference to the user.

### *2.5.1 Learning Bayesian Network from Data*

Training a Bayesian network has two parts. One is to learn the structure of the Bayesian Network from data. The second is to learn the conditional probability tables for nodes of Bayesian network from data. See Figure 2.8 [27].

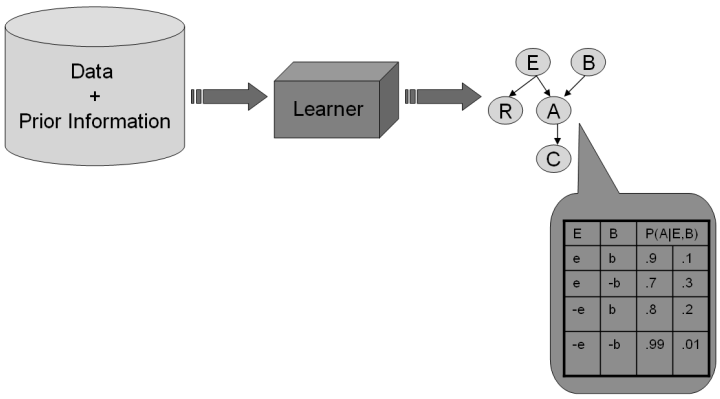



Figure 2.8: learning in Bayesian Networks

Moreover, Bayesian Network learning also has four situations.

- **Known structure, complete data**

  In this situation, the structure has been designed through other ways, and data does not contain missing values. Therefore, the learning task is to learn the CPT for every node.

- **Unknown structure, complete data**

  The learner has to learn the structure of the Bayesian network and estimate the CPT for nodes from the complete data.

- **Known structure, incomplete data**

  Although the structure has been specified, the data is not complete for some reason. The learner has to use the incomplete data to estimate the CPT for nodes.

- **Unknown structure, incomplete data**

  This situation is the most difficult one compared to the other three situations. The structure of the network and the CPT of every node has to be learned through incomplete data.

Similarly, there are many algorithms for learning in a Bayesian network based on different situations. We will not illustrate these algorithms here because the situation we are in is the first one, which has known structure and complete data. It is the simplest one compared to the others.

### *2.5.2 Bayesian Network Inference Applications*

A Bayesian network is a probabilistic graphical model which is based on the conditional independence concept among triplet of variables. Since the 1950s, Bayesian inference techniques have been used in the field of computerized pattern recognition techniques. Meanwhile, its applications also have been widely applied in artificial intelligent and expert systems.

A particular application of statistical classification was to develop an algorithm for identifying unsolicited bulk email spam [27]. For example, Bogofilter, SpamAssassin, InBox and Mozilla nowadays are applying Bayesian network to filter email spam.

In 2002, Juan Manuel Fernandez Luna [30] proposed a kind of information retrieval model. He used Bayesian Network, because the high performance of Bayesian network in the real problems characterized by uncertainly, and a suitable graph model that is able to represent and efficiently manipulate n-dimensional probabilities distributions.

In biology, Bayesian network applications also have been widely applied. For example, “the prediction of survival in patients with malignant skin melanoma” [31] is to perform a prediction of a survival after one, three or five years of being diagnosed as having malignant skin melanoma.

## 2.6 The Algorithms of Bayesian Network Inference

This section firstly briefly illustrates the network inference. After that it gives the methods of Bayesian network inference, and compares our approach to others.

### *2.6.1 Bayesian Network Inference*

The procedure of the network inference (network evaluation) is to update the probabilities of outcomes according to the relationships in the network and the

evidence known about the situation of the network. A Bayesian network is used, the end user considers recent events or observations as evidences. This network is instantiated by this information since the network has a variable of a state that is consistent with the evidence. When evidence is updated, the mathematical mechanics are invoked to update the probabilities of all the other variables, which are connected to the variable presenting the updated evidence. After updating, the probabilities represent the new levels of belief in all possible outcomes which was coded in the model.

Since the original probabilities, which are encoded in the network, are entered before any evidence is known about the situation, this kind of probabilities is called as prior probabilities. Moreover, after the evidence is captured, the computed probabilities are called posterior probabilities, because those probabilities represent the levels of belief that are computed in light of the new evidence [32].

### *2.6.2 Bayesian Network Inference Algorithm*

This subsection presents the algorithms implementing Bayesian network inference. Those algorithms are mainly classified into two classes: exact algorithms and approximate algorithms. Since we choose an exact method as our approach, we will focus on the exact algorithms.

As for the rationale of the exact algorithms, it is that the required quantities are calculated directly by complete enumeration of all hypotheses, and evaluation of their probabilities. On the other hand, the disadvantage of this kind of algorithms is that it just only can solve a few interesting problems which have a direct solution. However, this method is important as a tool for solving subtasks within larger problems.

Approximate algorithms are another class of the inference algorithm. Once the NP-hard complexity results are given, the main difficulty is how to design an efficient approximate algorithm. Comparing to the exact algorithms, the approximate algorithms are able to work for very large probabilistic models.

### *2.6.3 Exact BN Inference Algorithms*

For our approach, we choose the clique-tree propagation algorithm. In this subsection, we will compare this algorithm to other exact BN inference algorithms. Firstly, let's see the categories of the exact algorithms of BN inference. (See Figure 2.9). The Clustering algorithm, Loop cutset conditioning and Arc reversal are the three early exact inference algorithms.

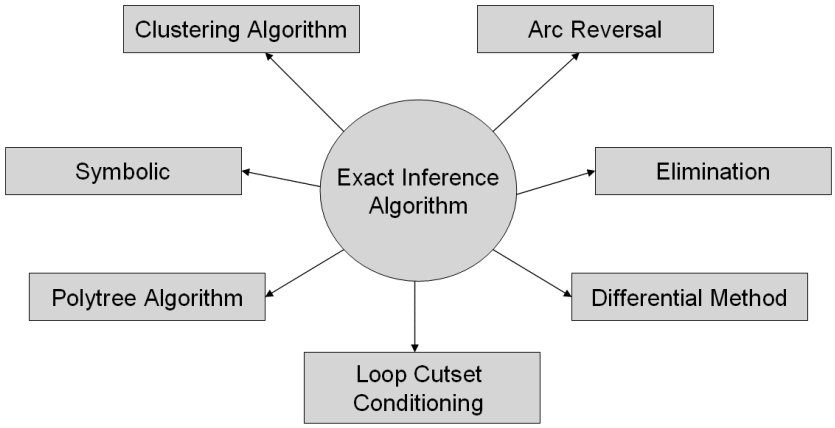



Figure 2.9: the categories of the exact inference algorithm

The Clique-tree propagation algorithm (clustering algorithm) was proposed by Lauritzen and Spiegelhalter in 1988 [32]. Meanwhile, the algorithm is the most popular exact BN inference algorithm. Firstly, a graphic transformation is implemented. It transforms a multiply connected Bayesian network, which is a directed acyclic graph, into a clique tree through clustering a triangulated moral graph of the underlying undirected graph. In the resultant graph, the "sepset" node contains the common variables of the "cluster" nodes which are connected with the sepset node. In the mean time, the conditional probability table (CPT) is generated for each cluster and sepset. Secondly, the algorithm initializes the tables of each cluster and each sepset. Finally, message propagation is performed over the whole cluster trees until a consistent cluster tree comes up.

- Advantages: it is able to perform an efficient work in sparse Bayesian network.
- Disadvantages: it is extremely slow when it works for the dense network.

Loop cutset conditioning algorithm was proposed by Pearl in 1986 [33]. It is an exact inference algorithm for multiply connected network. The algorithm performs a change of the connectivity of the network, and then a single connected network is generated. After that, the results of each instantiation are combined weighted by their prior probabilities.

- Advantages: it is able to work for multiply connected network.
- Disadvantages: the complexity of this algorithm is based on the size of the loop cutset, however, the loop cutset minimization problem is NP-hard.

Pearl also proposed Message propagation inference algorithm in 1983. The algorithm only works for polytrees.

Arc reversal, also called node reduction, was developed by Shachter [34]. It is one of the three early exact inference algorithms. The algorithm uses a sequence of operators to the network, which reverse the links using Bayesian rules. It keeps performing this process until the network is reduced to the query nodes with the evidence nodes as immediate predecessors.

There exist other exact inference algorithms. For example, Variable elimination algorithm sums out other variables in order to eliminate them one by one. Symbolic probabilistic inference regards probabilistic inference as a combinatorial optimization problem, the optimal factoring problem.

### *2.6.4 Approximate BN Inference Algorithms*

This kind of method can be subdivided into

1. Deterministic approximations, which include maximum likelihood, Laplace's method and variational method.
2. Monte Carlo methods, techniques in which random numbers play an integral part.

Approximate BN inference algorithms include stochastic simulation algorithms, model simplification methods, search-based methods and loopy belief propagation. See Figure 2.10.

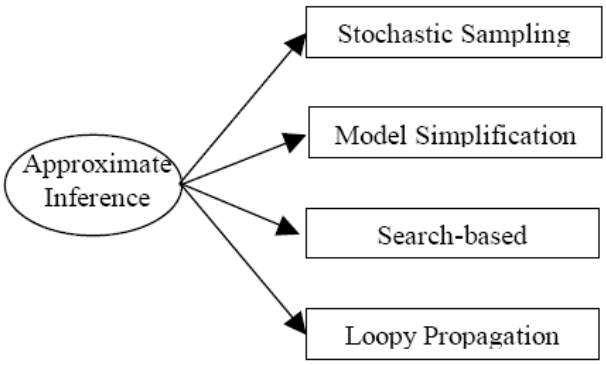



Figure 2.10: four algorithms for approximate inference

In the implementation part of this thesis, we will explain why we chose the clique-tree propagation algorithm according to our project's situation.

## 2.7 Summary

In this chapter, we briefly illustrated the past work related to our project. The following are some of the major points of the chapter:

- A number of electronic tags have been researched, some of which are similar to ours.
- Crowd simulation. In this section, we mainly discussed intelligent agents and their behaviour.
- Matchmaking. There are too many matchmaking methods to include individually, but we described the method we will use in our project, which is User-profile matchmaking.
- An comparison between Bayesian network and three main machine learning methods
- The definition of Bayesian network, and its applications.
- A discussion of Bayesian network inference algorithms.

Chapter 3

# THE SIMULATOR SYSTEM

The simulation system used in the Smart Badge project is the evaluation tool for the server system. Therefore, it has its own features and functionality making it suitable to the project. In this chapter, in Section 3.1, we first present the motivation for designing and implementing the simulation system. Section 3.2 illustrates the design of the simulation system and the functionality is discussed in Section 3.3. Finally, there are implementations (see Section 3.4) and summary sections (see Section 3.5).

## 3.1 Motivation

As the first chapter mentioned, Smart Badges could be used in several situations such as conferences, museums, or even cocktail parties. Consequently, when we are designing the smart badge, we must think about how to test and evaluate the product.

An ideal testing and evaluation of Smart Badges should be in a real conference environment where each attendee wears the Smart Badge device, while they are talking to each other and looking at posters on the wall. Researchers have to track them or follow them in order to collect enough data to train the Bayesian network, test the server system and achieve the evaluation's purposes. To obtain a general data set, we would normally have to ask several different attendee groups to repeat these same steps. However, a big problem with using real people and badges is that there are too many "confounding factors", i.e. factors that are outside your control. It could be difficult to draw any concrete conclusions about the system's behavior because the same set of circumstances (such as people, conversations and interactions) could not be duplicated. In our case, it was also impossible to run a study such as this as we didn't have access to a large number of real smart badge hardware, or people to wear the badges.

To overcome this problem we built a system that simulates a virtual conference to test and evaluate the server system. We use virtual agents as the attendees of the virtual conference. They have capability to look at the “posters” on the “wall”, and to talk to other agents. They are allowed to watch and talk. Meanwhile, the simulation system records the data that we want in real-time. Furthermore, we utilize network programming to implement simulated wireless communication between badges and the server.

## 3.2 Designing of the Simulation System

This section will explain how the simulation system was designed. In the first chapter, we mentioned that the simulation system will be used as an evaluation tool for the server system. Therefore, the goal is to create a virtual conference in which virtual agents move around randomly. In this section we explain the requirement of the project for the simulation system. After that, we describe the architecture for the simulation system, and its functionalities.

### *3.2.1 Requirements*

Our Smart Badge system involves a server machine and number of electronic badges. The server machine can communicate with badges through RF signals. Meanwhile, the badges can swap their data by using infrared sensors. The badges also can collect data from the environment they are in by using infrared sensors which are attached with some objects, such as posters on the wall. The badges send this data to the server machine. Consequently, the server can track every badge. The server also can figure out the interests of every wearer using an interest inference system. The badges can get the feedback from the server, such as the similarity between the interests of the wearer and the person that they’re talking to. This feedback is expected to augment the communication of people who attend this conference. The following figure shows the data flow.

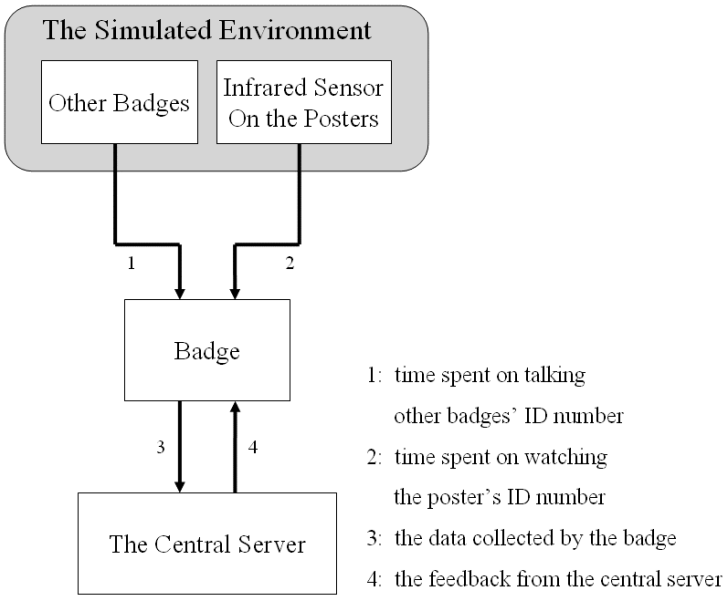



Figure 3.1: the data flow of the SmartBadge system

The simulation system should simulate human behaviours, and have a network connecting to send and receive data with the server. Therefore, the two main components of the simulation system are the behaviour simulator and the network connection.

### Requirements of the Simulator

The main point of the simulator software is to simulate the human behaviours as accurately as possible. Hence, firstly we require that there is a virtual environment which simulates the target real environment of a conference hall with several separated rooms.

We also need to populate the simulation space with virtual agents that are able to move, steer, avoid obstacles, stop, "talk" and "watch". Furthermore, every agent has the capability to record the time and positions. In the rest of the section we describe the following behaviours in more depth:

- **Moving**

  The virtual agent can move in the environment smoothly and continuously.

- **Steering**

The virtual agent can steer and rotate, when he encounters other agents or to avoid the obstacles. When he is performing these behaviors, he will accelerate or decelerate his body.

- **Avoidance of Obstacles**

The agent can avoid obstacles. That means it has a predictive capability. It can predict where the obstacles are, and determine which methods, such as stopping or steering, should be done to avoid obstacles.

- **Stopping**

The agent should be stopped when the user wants to stop it.

- **"Talking"**

"Talking" is a pause in the agent motion. When two agents encounter each other, if they are sufficiently interested in each other, then those two will stop and "talk" to each other; recording how long they spend talking.

- **"Watching"**

"Watching" is another type of pause in the agent motion. When an agent wants to look at a poster, the agent will move close to the virtual poster, and will stop there. The agent also records how long he spends "looking" at the poster.

The above contexts are the requirements for a simulator. The most important requirement is to really simulate human behavior in the real world in a desktop simulation environment.

### *3.2.2 Requirements of the network connection*

There are not too many requirements of the network connection. We need to create a connection, transferring the data, receive the data and closing the connection. The network protocol that is used was developed earlier by another student. This protocol takes security aspects into consideration.

### *3.2.3 Architecture*

To satisfy the system requirements, we designed the architecture shown in Figure 3.2.

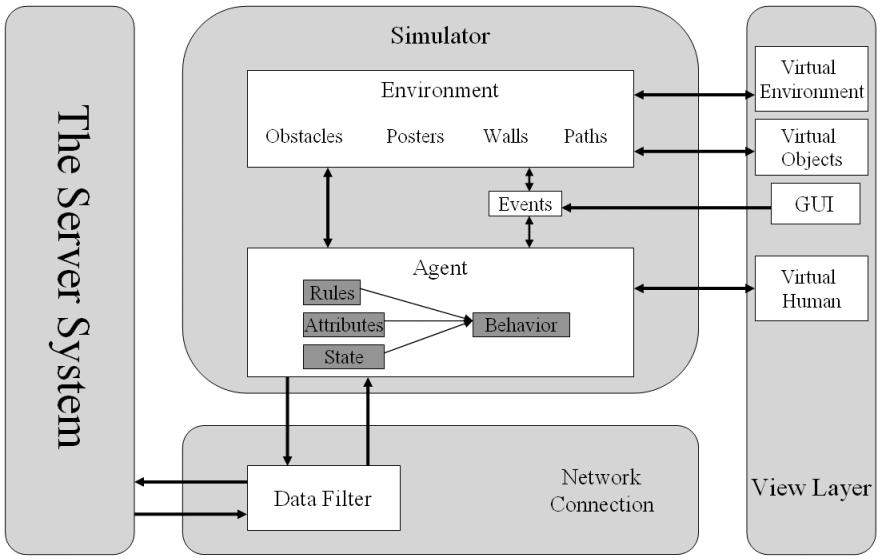



Figure 3.2: the architecture of the simulation system

## Agent

As you can see from Figure 3.2, the agents determine their behaviors by rules, attributes and their state. Generally, rules and attributes are designed by the system developer. For example, the rule can be “if the similarity between you and person you are talking is over 30, then the taking time should range from 4 minutes to 10 minutes”. The attributes can be “The interest level in Sport is 0.8”. On the other hand, we design several states for agents, such as “walking”, “talking” or “resting”. But we cannot assign a fixed state to an agent, because the state of the agent is determined by the situation the agent is in. For example, if an agent just finished talking with someone, and if his energy’s value is not high, then his state will be “Resting”.

Every agent has a virtual infrared sensor to perceive the change of environment. For example, if an agent is in range of a poster, then its virtual sensor will know this situation.

## Environment

The simulated conference space has some objects in it, including obstacles, posters, walls and paths. Obstacles are subdivided into static obstacles and dynamic obstacles.

A static obstacle is shown as a fixed virtual circle in the desktop simulation, while other agents are dynamic obstacles. Posters are placed on the virtual wall. Every poster has an interaction range, which means if the agent is in this range, it is able to look at the poster. We present five rooms in the environment. Each room denotes one different topic, such as Sport, Movies or Books. The simulation's main task is to keep agents inside of the rooms. Paths are used to lead the agents around the rooms. The agents can walk along with the path with different speeds and in different directions.

There is an interaction between the environment and the agents. The Agents can perceive the environment through the virtual sensors, and determine their next behavior. The behaviors of agents also impact the environment. For example, the movement of an agent can cause a potential obstacle to come up, and so affect other agent's behavior. Furthermore, the events which are from the user also impact both the agents and the environment. In our project, the user can pick any agent up and place it a new position or change the path using mouse input.

### Network Connection

Although the simulated network connection in our project does not have much functionality, it still plays an important role. It not only delivers the data from the agents to the server, but also has to be a data filter. This means it should choose the data to deal with. For example, if the agent wants to send data after talking with another agent, then the data filter just sends the time and the person ID, and who the agent was talking with to the server. On the other hand, if the agent just finished looking at a certain poster, then the data filter should send the time and the poster ID the agent was looking at, rather than the person ID. In addition to the feedback from the server, the data filter formats the feedback again and send it to the agents so that the agent can understand it. For example, if the feedback is "F/34/02", and the agent is in state of "talking", then the feedback will be formatted as "Hello, the similarity of the person you are talking to is 34".

**View Layer**

The user will see and control the simulator by the view layer. In the view layer, there are a virtual environment, such as walls, poster, obstacles and paths, the virtual human, and graphic user interface (GUI). The user can control the agent and change the environment by GUI. For example, the user can drag the virtual agent to any room in the simulated conference. See Figure 3.4.

## 3.3 Functionalities

The simulation system not only just presents a virtual conference to the user, but the user also can interact with the system through GUI. The functionalities the system provides include the following:

### *3.3.1 Control*

- **Drag the virtual agent**

  The user can drag the virtual agent to wherever he wants, including outside of the rooms. The user uses the mouse right button to select an agent and drag it.

- **Send a "Watch" demand**

  The user can send a "watch" demand to the simulation system. After that, the agents are able to look at the posters.

- **Send a "Talk" demand**

  The user can send a "talk" demand to the simulation system. After that, the agents are able to talk to each other.

- **Send a "Stop" demand**

  The user can demand all agents stop their activities.

### *3.3.2 View*

- **Show the path**

  The user can choose the option to see the path along which the agents walk.

- **Show the obstacle avoidance**

The user can choose the option to see how the agents avoid the potential obstacles.

- **Show the view range of the agents**

The user can choose the option to see the view range of the agents.

### *3.3.3 Configure*

- **Connect to the server**

The user can determine when the simulation system connects to the server. For this the user needs to know the IP of the server machine.

- **Change the number of the agents**

The user can change the number of the agents from 0 to 50.

- **Change the view range and the view angle of the agents**

The user can choose a view range from 1 meter, 2 meters and 3 meters, and change the angle of the view, from 0 to 180 degrees. Since the range of infrared is about 3 meters and the angle is about 30 degrees, the default settings are 3 meters and 60 degrees.

Figure 3.3 below shows the functionalities of the simulation system.

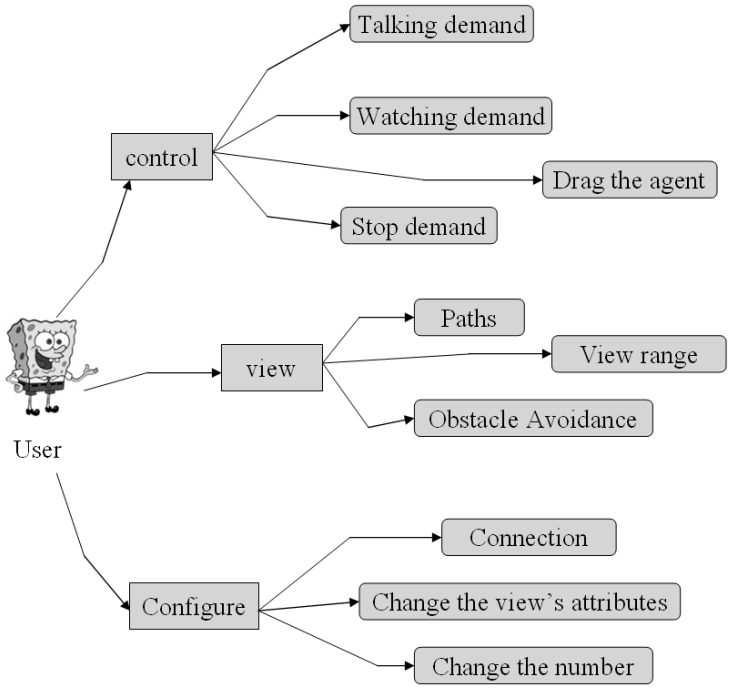



Figure 3.3: the functionalities of the simulation system

## 3.4 Implementation

This section will describe the implementation of the simulation system. Firstly, we present the graphic user interface which is developed by MFC library. The next section mainly introduces building of the intelligent agents and building of the environment. We designed the method to determine the time duration of talk. Basic behavior will be developed using the OpenSteer library.

### *3.4.1 Graphic User Interface*

According to the requirements and the functionalities of the simulation system, there is an interaction between the system and the user. Therefore, the main tasks of the GUI are not only to represent the view of the simulator, but also to provide the interface to the user so that they can control the objects or choose the options.

We used C++, the Microsoft Foundation Class (MFC), and OpenGL to implement those tasks. The MFC library can be used to provide the application with a standard Microsoft windows style, and a flexible framework. OpenGL is a well-known graphics library which can be used to present the rotation, transformation and movement of agents. Consequently, OpenGL will be the graphic development tool used in our project.

The purpose of the simulation is not to show the detail of agents and other objects, but to simulate the agents' behavior. Accordingly, it is not necessary to use a full 3-D environment in order to make a movement pattern for each agent. As a result, we choose a 2-D view for the presentation of the simulation output. In the simulation (Figure 3.4) an agent unit will be presented by a particle that shows the body of the agent and a transparent sector that displays the range of the agent's view and badge sensors (Figure 3.7). Yellow lines on the screen will represent walls and doors of rooms. The middle circle is a big obstacle which can let the agents walk along with path very well. There are 20 light grey half circles which are placed along with walls. Those grey half circles denote the poster's detecting range, and the centre point of the half circle denotes one poster. There also is a path, with which the agents can walk along (See Figure 3.8).

To make the simulation more readable, when the mouse pointer clicks at an agent, the agent will be coloured orange. Not only that, but the simulation will also allow the user to drag and drop the agent. The user may put the agent anywhere he wants by dragging and dropping the agent.

Apart from the presentation of the simulation, there is another view on the right side of the GUI for displaying the agent's profile. Every agent has lots of information, such as name, ID, the company, and so on, so the checkbox and drop-editor will be applied to present the agent profile. It is most important that the checkbox and the drop-editor can be pre-assigned with a set of fixed values. (See Figure 3.4)

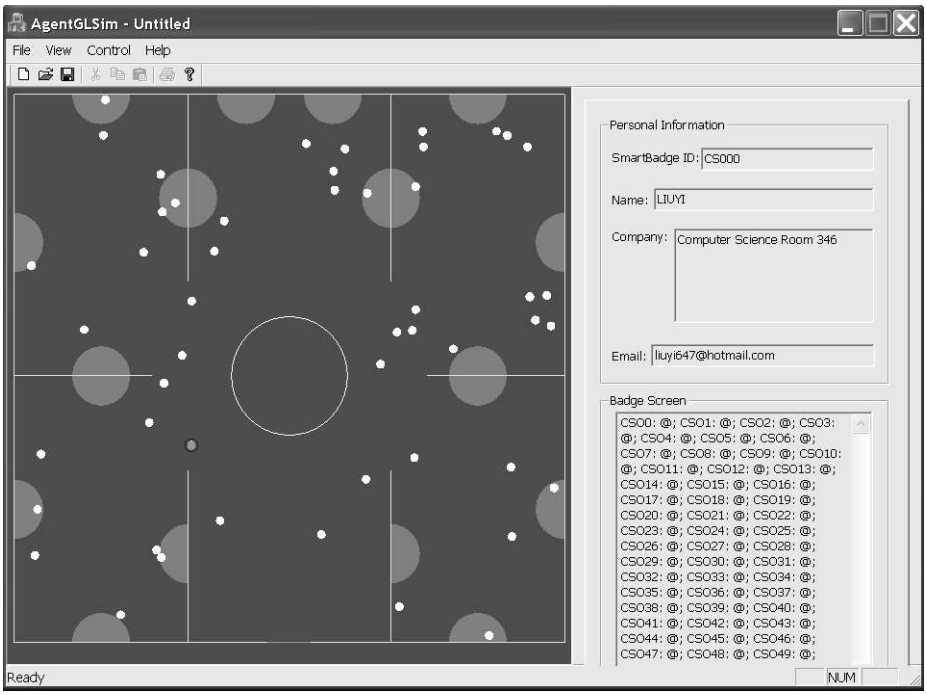



Figure 3.4: the GUI of the Simulation system

The simulation application will allow the user to more easily, quickly and effectively obtain the information he wants. The user can choose the options in menu. The options include the functionalities mentioned before. For example, if the user chooses the menu -> “View” ->”Badge detecting Range”, then you can see the result from Figure 3.7. The green sectors denote the range and angle of the badge detecting range. If the user chooses the menu -> “View” ->”Path”, then you can see the result from Figure 3.8. The red line represents the path.

To control the agents, the user can choose “Talking”, “Watching” and “All stop”. Figure 3.5 shows the control menu. The user also can configure the setting of the simulator and the agents using the “Option” item of the menu. The dialogs of “Option” and “Connection” are shown in Figure 3.6.


Figure 3.5: the control menu

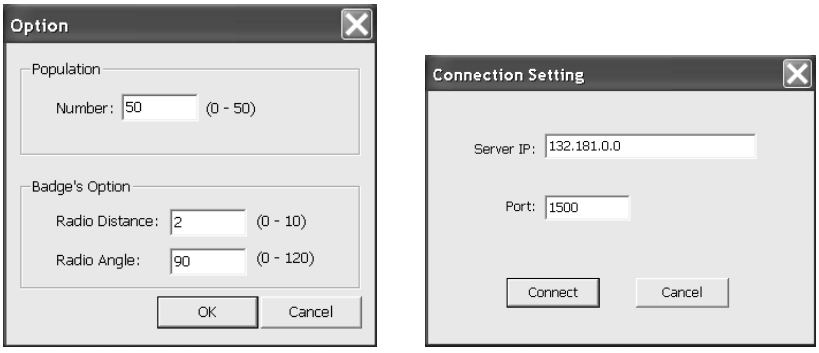



Figure 3.6: the Option dialog and the connection dialog

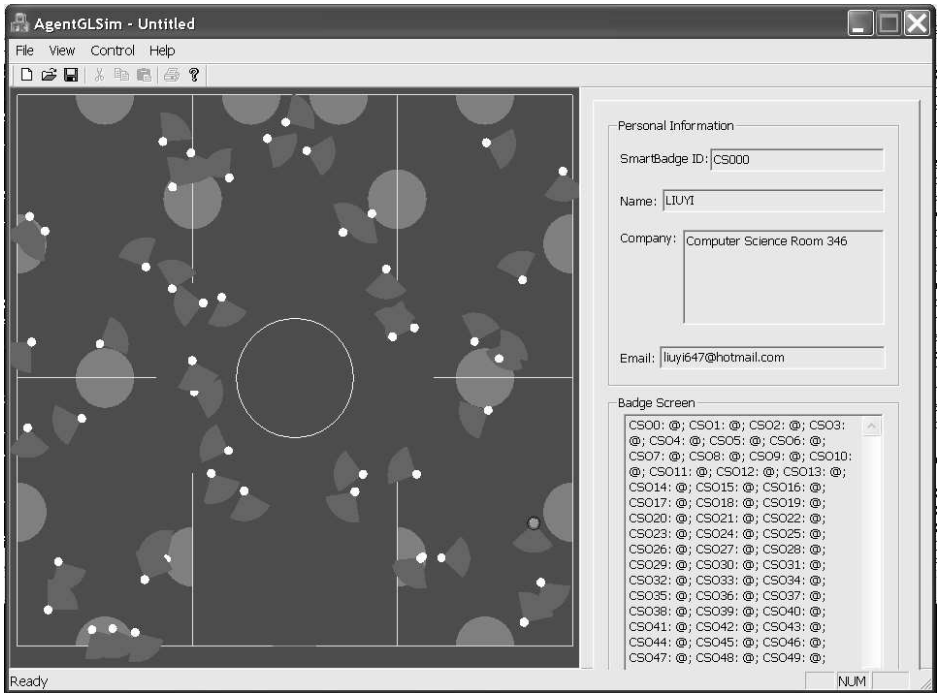



Figure 3.7: show the badge detecting range

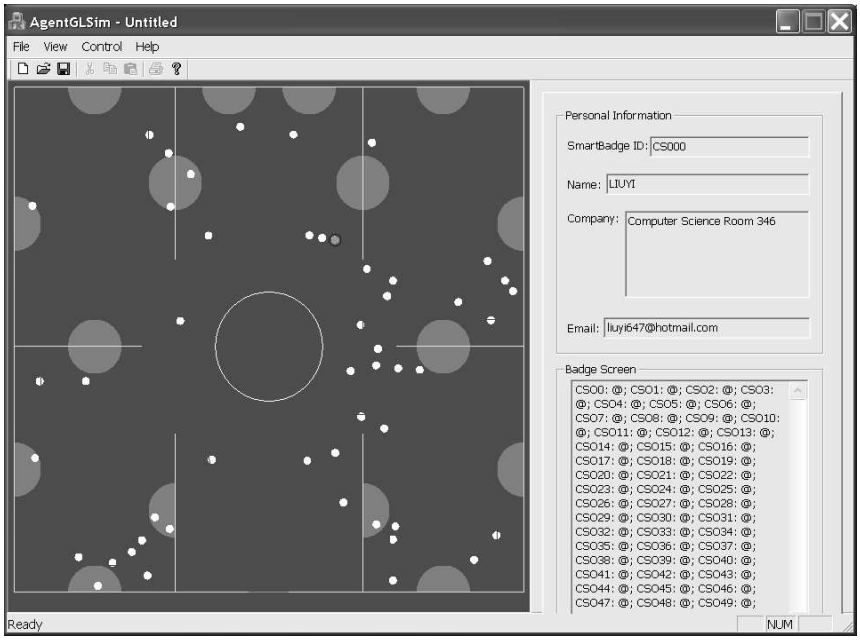



Figure 3.8: show the path of the simulator

### *3.4.2 The Intelligent Agents*

Every particle denotes a virtual human (Agent). If we do not assign anything to those, they just move around randomly without any purpose. In our project, we not only assigned the registration information to them, but we also designed States Mechanism for them.

#### Registration Information

The registration information of agents in our project was artificial. We designed data for 50 agents. The registration information includes an agent's name, ID, email, and company name. Of these, the most important is the agent's ID. We call it Person ID, because it is unique, just like the identity card of a person. Person ID is used to identify the agent in the simulation system and the server system. The registration information also includes the interest table which presents agent's interests and the level corresponding to every interest. The interest table will be used to determine the time spent looking at posters and talking. We will explain how to determine this later. In

short, all registration information will be loaded at the beginning of the simulation from a file of AgentInfo.dat. The file is shown in Appendix A.

### States Mechanism

We design a State Mechanism for the agents. The agents can determine the behaviors through the state they are in. The state can also be changed into another state. Figure 3.9 shows the state mechanism.

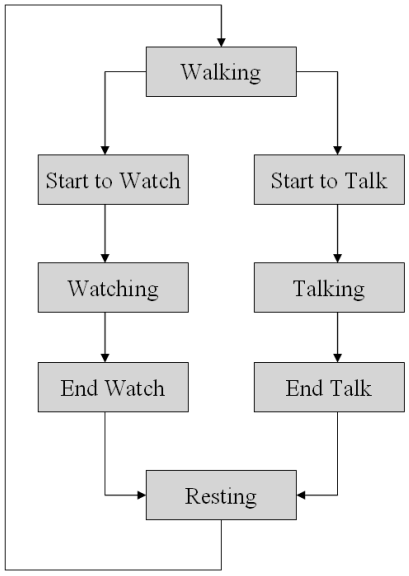



Figure 3.9: the State Mechanism

At the beginning of the simulation, all agents will be in the state “walking”. They will encounter different situations, such as meeting someone, or moving close to a poster. Their state will then be changed according to those different situations. For example, if an agent encounters another one, they probably spend some time talking. Their states are both changed into the state “start to talk”. In this state, the agent will send a message which includes his Person ID, and Person ID of another one to the server. After that, he will get the feedback which is the similarity value of their interests from the server. The state will be transferred into the state “talking”. In this state, the agents will record the time they spend talking. When they finish talking, the state will be changed into the state “end talk”. In this state, the agent will send a message which includes the time and Person ID to the server. At this time, there is no feedback from

the server. After that, the state will be in the state of “Resting”. How long the agent is in “resting” is determined by its energy value. We will write more about the energy value of the agent later. After this, the state will be changed back to “Walking”. Similarly, the same situation will happen when the agents are going to look at the posters.

So the interaction sequence is the following:

- **Walking**

  The agent is ready to talk and watch.

- **Start to watch**

  The agent is going to look at a poster, and sends the message to the server. Meanwhile he also receives feedback from the server.

- **Watching**

  The agent is looking at the poster, and records the time.

- **End watch**

  The agent just finished looking at a poster, and sends the message to the server. There is no feedback.

- **Start to talk**

  The agent is going to talk with another one, and sends the message to the server. Meanwhile he also receives a feedback.

- **Talking**

  The agent is talking with another agent, and records the time.

- **End talk**

  The agent just finished talking, and sent the message to the server. There is no feedback.

- **Resting**

  The agent takes a moment rest, and then changes back to the state “walking”.

### Energy Value

For simulating real human behaviors, we built an energy value computing method which can assign an energy value denoting the agent’s energy to every agent. How long the agent is in the state “resting” after talking and watching is determined by this energy value.

The energy value will be randomly assigned at the initialization of the simulation system. Every agent has different energy value ranging from 2 minutes to 7 minutes which determines how long they can interact for. For example, if an agent's energy value is 5 minutes, then it has to take 5 minutes break before it starts talking or looking at posters.

### How to Determine the Talking Time

For this point, we deal with two different situations. One is there is no matchmaker support. This means the badge does not provide feedback so that the wearer does not know what the common interests the person he is talking to has. Therefore, it is just like two strangers meeting together. In the real world, if two strangers encounter together, they will probably talk only a short time. They may talk longer if they are interested in similar topics. They are all uncertain because they do not know each other before they start to talk. Therefore, we use a probabilistic method to determine the talking time for this situation. We simply divide the time into three states which are short, middle and long. We say when two strangers encounter and talk with each other the talk time could be short, middle and long, and the probability of those three states should be same (see Table 3.1). In Table 3.1, there are two columns which are "time talking" and "probabilities". For example, if two strangers encounter, they have same percentage chance to talk for a long, middle or short time. They could just say hello or they could talk about the weather something, or they could find out their common interests.

| **Time talking** | **Probabilities** |
|---|---|
| *Short* | 0.333 |
| *Middle* | 0.333 |
| *Long* | 0.334 |

Table 3.1: the probability for talking time without matchmaker support

On the other hand, there exists matchmaker support. This means that before two strangers encounter each other, the matchmaker will announce to them their common interests. That is to say, they know about each other before talking, and they should talk a long time if they have many common interests. Not all people are talkative, so they would probably not like to talk a long time even when they know another one's

interests. To make it simple, we make an assumption that all agents are talkative. At this time, we determine the talking time according to the similarity of two agents. If an agent meets someone with a high similarity, then they must spend a long time talking. Therefore, the similarity may become a standard to determine the talking time. Before that, we must make the second assumption which is that the matchmaker should be 100% accurate. We still use a probabilistic method to do that. The similarity can be classified into High, normal and low (see Table 3.2). In this table, we can see there are three columns which are similarity, time on talking and probabilities. For example, when the similarity is high, the two agents have an 80% percent chance to talk for a long time, and just 10% percent to talk for a short or middle time.

| Similarity | Time on Talking | Probabilities |
|---|---|---|
| High | Short | 0.10 |
| High | Middle | 0.10 |
| High | Long | 0.80 |
| Normal | Short | 0.10 |
| Normal | Middle | 0.80 |
| Normal | Long | 0.10 |
| Low | Short | 0.80 |
| Low | Middle | 0.10 |
| Low | Long | 0.10 |

Table 3.2: the probabilities of talking time with matchmaker support

As for time spent by agents looking at posters, we will use two different ways to determine this as explained in Chapter 6.

### *3.4.3 Basic Behaviors*

The basic behaviors of the agents are moving, steering, and obstacle avoidance. To implement these behaviors the OpenSteer library was used.

OpenSteer [35] is an open source library of components for building steering behaviors for autonomous characters in games and other kinds of multi-agent simulations. These agents may represent characters (humans, animals, alien creatures), vehicles (cars, planes, spaceships) or other kinds of mobile agents. OpenSteer provides a toolkit of steering behaviors, defined in terms of an abstract mobile agent.

In our simulator, the main agent's behaviors are path following, and avoiding neighbors and obstacles. Hence, we chose those functions related with these behaviors from the OpenSteer library.

### Path Following Behavior

```
Vec3 steerToFollowPath (const int direction,
                        const float predictionTime,
                        Pathway& path);
Vec3 steerToStayOnPath (const float predictionTime, Pathway&
path);
```

**Functionality:**

It is to provide a steering force to follow a given path. `steerToStayOnPath:` it is to keep the vehicle on the path. `steerToFollowPath:` it is to provide *directed path following* where the vehicle both stays on the path and heads in a given direction along the path.

**Argument:**

`direction:` it should be either +1 or -1.

`path`: it defines a *tube* in terms of a *spine* and a *radius*, the goal is to keep a vehicle headed toward a point inside that tube.

`predictionTime:` it is to give a prediction of the agent's future position in a prediction time. Steering is determined based on the prediction.

**Return Value:**

If that predicted position is inside the pathway (and in the case of directed path following, is headed in the correct direction) this function returns a zero vector value. Otherwise it steers toward a point on the path.

### Obstacle Avoidance Behavior

```
Vec3 steerToAvoidObstacle (const float minTimeToCollision,
                           const Obstacle& obstacle);
```

```
Vec3 steerToAvoidObstacles (const float minTimeToCollision,
                            const ObstacleGroup& obstacles);
```

**Functionality:**

It is to provide a steering force to avoid the obstacles. The obstacles could be a single one or a group of obstacles. “The purely lateral steering force will turn our vehicle towards a silhouette edge of the obstacle” [35]

**Argument:**

`minTimeToCollision:` it is a specified time value which is min time to collide at the agent’s current velocity.

`Obstacle:` it is an argument to specify an obstacle.

`Obstacles:` it is an argument to specify a group of obstacles.

**Return Value:**

If there is no avoidance required, then the function returns a zero vector value.

### Unaligned Collision Avoidance Behavior

```
Vec3 steerToAvoidNeighbors (const float minTimeToCollision,
                            const AVGroup& others);
```

**Functionality:**

It is to provide a steering force to avoid colliding with other close agents moving in unconstrained directions. It determines which other agent is the first collision we should avoid.

**Argument:**

`minTimeToCollision:` the same with above one.

`Others:` a group of agent objects.

**Return Value:**

If there is no avoidance required, then the function returns a zero vector value.

## *3.4.4 The Network Connection and Data Filter*

For the network communication, we used the Winsock library [36] because it is easy to integrate this into an MFC project. A Winsock object was developed which included opening a Winsock connection, initiating the Winsock connection, a data filter, sending

message and closing the connection. Every agent can call it and pass his message into the object as the input parameter.

Meanwhile, the simulation has a detecting program which can detect the feedback from the server. Once it detects a feedback, it will know who the destination is, and format and deliver the feedback to the agent. We have made an example of data filter, so here we do not repeat it.

## 3.5 Summary

In this chapter, we report on how we implemented the simulation system. The main points are as following:

- **Motivation**

  We explained why we need a simulator.

- **Requirements**

  We described the requirements for the simulation. To satisfy those requirements, we designed the architecture for the system.

- **Designing of the simulation system**

  We worked out the architecture of the system, and describe every part of the architecture in detail.

- **Functionalities**

  What functions the user can choose have been placed in this part.

- **Implementation**

  We reported how to implement the GUI and how to determine three time problems. Finally, we described the implementation of network connection.

Chapter 4

# THE CENTRAL SERVER SYSTEM

So far we have described the simulation system. As the core part of this project, the server system involves a matchmaker and an inference module. In this chapter, we analyze the requirements for the server system, and then describe the architecture of the server system. Finally, we report on how to construct the inference module and the matchmaker respectively.

## 4.1 Requirements

Before designing the architecture of the server, we have to analyze how the server should perform. First, we can describe the real conference environment in which Smart badges could be used. The attendee is wearing a badge, looking at posters on the wall, and talking to other conference attendees. His badge tracks his behaviors. When he is looking at a poster, the badge can record the time spent reading the poster and the poster's ID. After that, the badge will send the record to the server. Then, the server can provide feedback to him about his interest level in the poster. The input data of the server system is the time spent on looking at the poster, and the ID of poster. Currently, we use this data as the input to the inference module. Meanwhile, we use the result of the inference module as the input data to the matchmaker.

There are two things required of the inference module. One of them is that it is able to infer the attendee's interests according to the posters he has read and how long he has spent looking at those posters. We require that the inference module can infer the attendee interests in real time. For example, when you have just finished reading a poster, the inference module should be able to work out how much you are interested in the topic the poster is presenting. So, the inference model should infer the conference attendee interests correctly according to the limited information about their behavior.

For the matchmaker, the requirements are similar to those of the inference module. Firstly, the matchmaker should figure out the correct similarity of interests for attendees who have just met together. Secondly, the matchmaker should send the resultant feedback in real time. Finally, an extra requirement is that the computing time should be as short as possible because it is necessary for attendees that they would get the feedback near the beginning of their conversation.

## 4.2 Architecture

We designed the architecture for the server system shown in Figure 4.1.

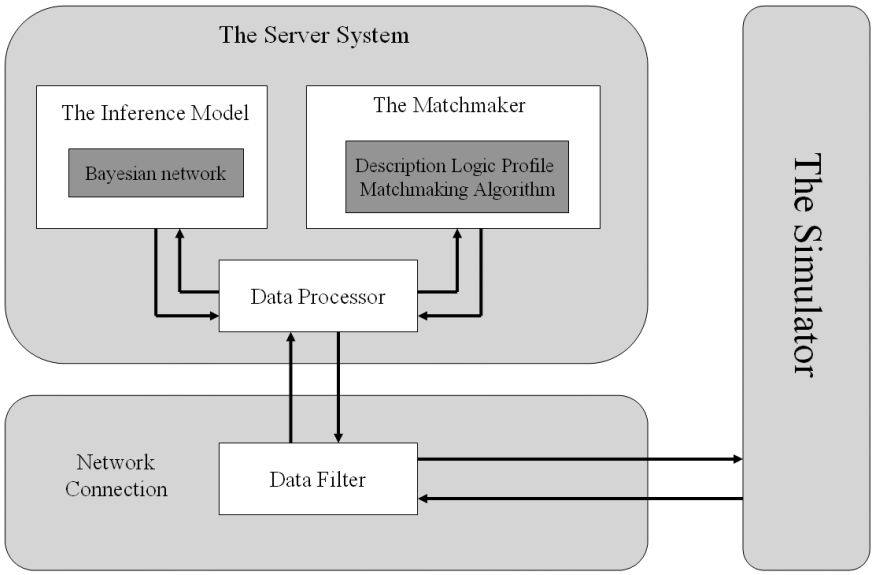



Figure 4.1: the architecture of the server system

The matchmaker is in charge of computing the agents' interest similarity using the Description Logic profile matchmaking algorithm, and then passing the result to the data processor. The inference module can infer the interests of badge wearer in real time from the limited information from the badge sensors, using a Bayesian network, and this result is also passed to the data processor.

When the data processor obtains the result of the matchmaker, it can work out what the relationship is of the two agents that have been matched. After that, it determines whether their common ideas and interests should be shared. For example, if the relationship is friendly, then they should have more common interests than in a neutral relationship. Therefore, the data processor would group those common interests and format them as a kind of representation. Finally, the data processor passes these to the network connection part so that the badge wearer can clearly know what the other attendee likes and dislikes. Sharing interests in this way can efficiently enhance human to human communication.

When the data processor has the result of the inference module, it can work out what the new agent's interest is and update the old interest table. After that, the data processor will search some related information about the interest and give feedback to the badge and the agent who wears it. For example, if the server system knows you are interested in Sport, then the badge could show you where other objects about Sport are. Or when you are looking at a poster about Sport, the badge could provide more detailed information about this poster because the system knows you like it.

In the next several sections, we will first illustrate the implementation of the inference module, then we will explain how the Description Logic represents the user's profile, and the matchmaking algorithm used.

## 4.3 Preparation for the Implementation

### *4.3.1 Interest Hierarchy*

The inference module and the matchmaker both use the concept of an interest hierarchy. To explain this, we first present the interests designed for the virtual agents. See Table 4.1. As you can see, there are 20 kind of interests defined in advance. We use keywords to represent them. Every keyword corresponds to a code, for example the

code for hiking is 2. In the program, we will use the codes to replace the keyword of interest.

| CODE | INTEREST |
|---|---|
| 1 | Sport |
| 2 | Hiking |
| 3 | Basketball |
| 4 | Swimming |
| 5 | Movies |
| 6 | Martial movies |
| 7 | Horrific movies |
| 8 | Romantic movies |
| 9 | Books |
| 10 | Magazine |
| 11 | Novel |
| 12 | Newspaper |
| 13 | IT |
| 14 | AI |
| 15 | Network |
| 16 | Computer Graphic |
| 17 | Idols |
| 18 | Sport stars |
| 19 | Movie stars |
| 20 | Other idols |

Table 4.1: the interest designed for the virtual agents

In this list, there are interests like sport, movies, books, computer technology and idols. We call those keywords *Father* keywords. On the other hand, there are other keywords, such as hiking, horror movies, novels, AI and sport stars which are sub-classes of these father keywords. For example, hiking is a kind of sport or horror movies are a kind of movies. We call those keywords *Child* keywords. Using father and child keywords, we can generate an interest hierarchy. See Figure 4.2. The interest hierarchy will be combined with Description logic to create the agent's profile. We will illustrate the use of an interest hierarchy in Section 4.5.

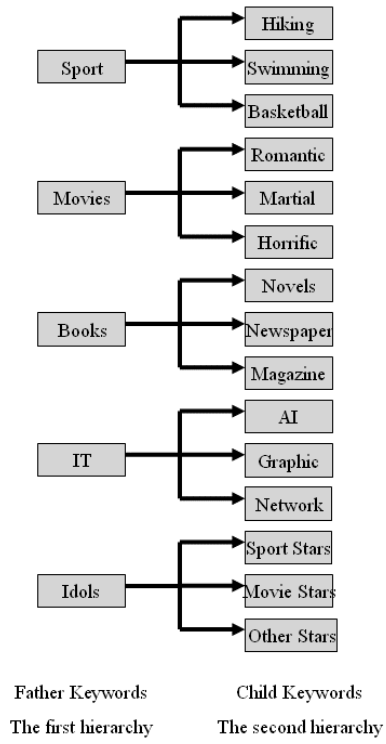



Figure 4.2: the interest hierarchy

### Interest Group

The interest hierarchy will mainly be used in the matchmaker algorithm. But the sub-concept (child keywords) will be used in the inference module. We group the interests as five groups according to the interest hierarchy. For example, hiking, basketball and swimming belong to sport, so we group the three interests and sport as one group. We assign a code to each keyword, since it is easy to be understood by the program. For example the code of ‘sport’ is 1, The interest groups will be used in the inference model. The state of groups will be nodes of Bayesian network. If a keyword in the groups is relevant to the keyword the agent is looking at, then it suggests the agent probably has interest about the current keyword. See Table 4.2. The use of the interest groups will be presented in Section 4.4.

| Group Code | Interests | Code |
|---|---|---|
| Group 1 | Sport | 1 |
| | Hiking | 2 |
| | Basketball | 3 |
| | Swimming | 4 |
| Group 2 | Movies | 5 |
| | Martial movies | 6 |
| | Horrific movies | 7 |
| | Romantic movies | 8 |
| Group 3 | Books | 9 |
| | Magazine | 10 |
| | Novel | 11 |
| | Newspaper | 12 |
| Group 4 | IT | 13 |
| | AI | 14 |
| | Network | 15 |
| | Computer Graphic | 16 |
| Group 5 | Idols | 17 |
| | Sport stars | 18 |
| | Movie stars | 19 |
| | Other idols | 20 |

Table 4.2: the Interest Groups

### *4.3.2 Assumptions used to model the Bayesian Network*

A Bayesian network is used to infer the agent's interests. To model the Bayesian network we make several assumptions:

- The level of the interest in a $Keyword$ is represented by a static node rather than a dynamic node. It can change but it is not a variable that evolves over time. What this means is that if the agent interest about a topic has been inferred, then it cannot be changed even if the agent spends a shorter amount of time looking at another poster about the same topic. If the level of the interest is dynamically changing, it can complicate the task. For example, when the interest of a keyword is changed from "interesting" into "boring", the interest level of the other keywords which are inferred based on the original level of the keyword have to be changed. This can be a huge computation when there are hundreds of keywords.
- To initially simplify the task, I have selected a subset of 20 representative posters in the simulator. Each poster represents the information about only one

keyword. For example, if the topic of a poster is basketball, then all information represented by the poster is about the basketball. Moreover, we have created five rooms in the simulator each representing one interest group which has been described in Section 4.3.1. For example, if we assume the first room contained information about Sport, then in the room there exist four posters which contain different sport keywords respectively. Actually, the central server is required to deal with hundreds even thousands of poster keywords. However, it will take a long time to finish the inference for all keywords for all agents.

- For the agents, we assume that each agent just looks at each keyword once. We consider the time spent looking at one keyword in the simulator as the sum of time spent on the keyword in real world. For example, if an agent spent 3 minutes on a keyword in the simulator, then we could say in real world he spent 1 minute on the poster containing the keyword, and then maybe he would spend another 2 minutes on other posters which contain the same keyword. In real life, there should be many posters for each keyword. But in the simulation, we suppose just one poster represents one keyword.
- We assume that each agent would not talk with the same agent more than once.

Once the basic modeling assumptions have been made various choices are still available when designing or modeling a Bayesian network in practice. It is clear that the model needs to be some kind of dynamic Bayesian network, as estimating the agent's interest is a procedure that happens in time while watching the posters. Therefore we need to model a procedure with 20 time slices, as there are 20 poster objects in the simulator. In addition to this, the actual placement of these objects on the museum floor provides some constraints on the model architecture. Therefore the geography of the objects in the simulator needs to be reflected in the topology of the modeling network. The choices available in modeling this problem are:

- the topology of the network
- the states per node

## 4.4 The Implementation of the Inference Module

The inference module mainly consists of a Bayesian network. We have described background knowledge about Bayesian networks in Chapter 2. Therefore, we will just describe how our Bayesian network was created. After that, we will transfer the network into the junction tree based upon the steps described in Chapter 2. The inference procedure has already been described in Chapter 2, and so we will not repeat that.

### *4.4.1 Bayesian Network Creation*

We will create the Bayesian network used in the inference module using several creation steps. To do this we need to specify the topology of the network and the states per node. We consider the creation of Bayesian network based on just one time slice. After that, we will model the whole Bayesian network according to the result of that one time slice.

#### Create a Set of Variables

The first step is to create a set of variables representing the distinct elements of the situation being modeled. In our real situation, we look forward to inferring the agent interests based on the time spent looking at posters and whether there is related keyword in his or her interest group or not. Therefore, the time spent looking at posters can be a variable and the type of interest group also can be a variable. How much interest the agent has about a keyword can be a variable. We call the three variables:

*Duration* (D): how long the agent stays looking at each poster

*Keyword* (KW): the level of interest in the poster the agent is closest to by

*InterestGroup* (IG): which shows if there is related keyword to the current

Now we can obtain the following result (Figure 4.3).


Figure 4.3: the variables of the Bayesian network

### States per Node

Next we need to define the states per node. For each variable, this step defines the set of outcomes or states that each can have. This set is referred to in the mathematical literature as "mutually exclusive and exhaustive," meaning that it must cover all possibilities for the variable, and that no important distinctions are shared between states.

Our Bayesian network has three kinds of variables(nodes), which are IG, KW and D. IG denotes $InterestGroup$ , which has two states, $Yes$ and $No$ . State $Yes$ means there is at least one related keyword in the interest group. State $No$ means there is no any related keyword in the interest group.

The interest group of every agent will be initially empty. As the number of interests increase, the program will update the interest group according to the interest hierarchy. For example, your interest group is empty at beginning and after a while, the program has updated your interest group as follows:

| **Your interest group** |
|---|
| Football |
| Horror movie |

After you have just finished looking at a poster about basketball the program will compute how much you are interested in Basketball. It will first take a look at your interest group to see whether there is a related keyword in it. There is one related

keyword which is Football. Therefore, the state of IG is Yes. If the program figures out the state of KW is Interesting, then it will update the interest group as follows:

| **Your interest group** |
|---|
| Football |
| Horror movie |
| Basketball |

Thus, the interest group is not static, but is dynamic due to the running of the inference model.

KW denotes *Keyword*, which has *Interesting*, *Neutral* and *Boring* states. The state *Interesting* means that the agent is interested in a certain topic. The state *Neutral* means that the agent is neutrally interested in a certain topic. The state *Boring* means the agent is not interested about the topic. Concretely, in this application, the states of this node are determined by the interest's level of every keyword. Thus, we suppose *Interesting* state's corresponding interest level is 0.6 - 0.9. *Neutral* 's interest level is 0.3 – 0.6. *Boring* 's interest level is 0 – 0.3.

D denotes *Duration*, which also has three states. They are *Long*, *Middle* and *Short*. The state *Long* means the agent spent a long time looking at a certain poster. For example, we could suppose *Long* state is over 6 minutes in the simulator. The state *Middle* means the agent spent an average amount of time on a certain poster, like 4 to 6 minutes in the simulator. The state *Short* means the agent spent a short time on a certain object, like 0 to 3 minutes in the simulator. All of the states are shown in the following figure.

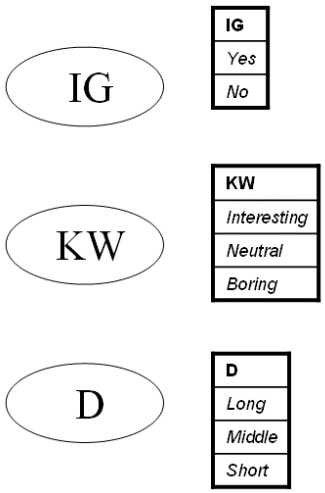



Figure 4.4: the variables with states of the Bayesian network

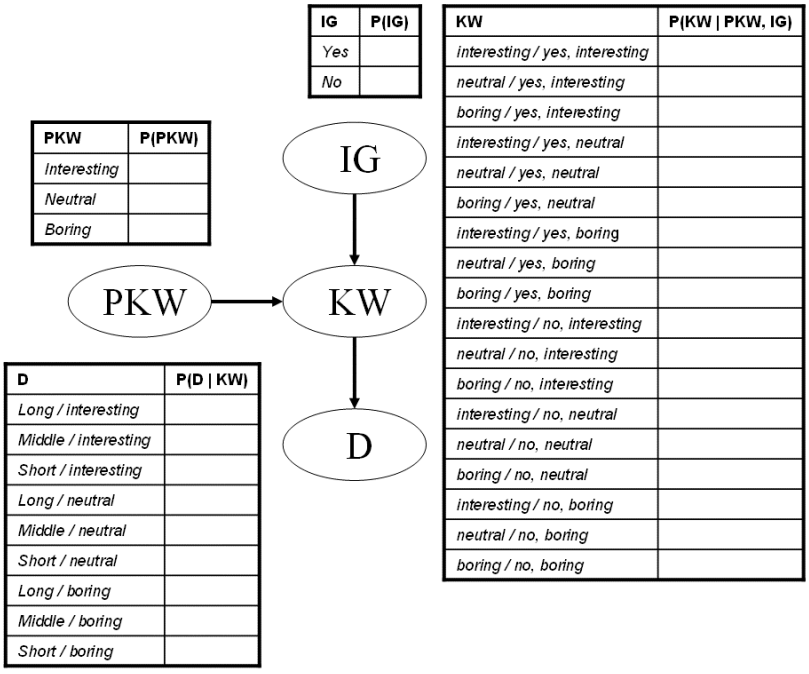
### Casual Relationship

The next step is to establish the causal dependency relationships between the variables. This involves creating arcs (lines with arrowheads) leading from the parent variable to the child variable.

What causal relationships exist in our situation? Firstly, we recall that the task is to infer how much the agent is interested in a certain keyword according to the time spent on this keyword and whether there are related keywords or not in his interest group. The result is that there are three events which can be represented by nodes IG, KW and D.

Secondly, we analyze the causal relationship. We say that the current status of the agent's interest group can affect the interest level in the current keyword. Therefore, there should be an arrow-line from node IG to node KW. Moreover, how much the agent is interested in the poster object must affect how long the agent is willing to spend looking at the poster. So there should be an arrow-line from node KW to node D. Finally, the agent's past history can affect how much interest in the current poster, so there should be an arrow-link from the preceding keyword to the current keyword.

Consequently, we add a variable called $PriorKeyword$ (PKW). PKW denotes the last keyword looked at. For example, if you just watched a poster about Movies, and then you transferred your attention to a poster about Books, then the keyword of Movies is the preceding keyword of the keyword of Books. The states of PKW are the same as the states of KW. Finally, we can get the Bayesian network shown in Figure 4.5.




| IG | P(IG) |
|---|---|
| Yes | |
| No | |

| PKW | P(PKW) |
|---|---|
| Interesting | |
| Neutral | |
| Boring | |

| KW | P(KW \| PKW, IG) |
|---|---|
| interesting / yes, interesting | |
| neutral / yes, interesting | |
| boring / yes, interesting | |
| interesting / yes, neutral | |
| neutral / yes, neutral | |
| boring / yes, neutral | |
| interesting / yes, boring | |
| neutral / yes, boring | |
| boring / yes, boring | |
| interesting / no, interesting | |
| neutral / no, interesting | |
| boring / no, interesting | |
| interesting / no, neutral | |
| neutral / no, neutral | |
| boring / no, neutral | |
| interesting / no, boring | |
| neutral / no, boring | |
| boring / no, boring | |

| D | P(D \| KW) |
|---|---|
| Long / interesting | |
| Middle / interesting | |
| Short / interesting | |
| Long / neutral | |
| Middle / neutral | |
| Short / neutral | |
| Long / boring | |
| Middle / boring | |
| Short / boring | |

Figure 4.5: the Bayesian network

### The Whole Bayesian Network

The preceding model of the Bayesian network is based on one time slice. If we consider 20 time slices together, then the model will be extended like a chain. See Figure 4.6. In this figure, IG still denotes the Interest group and KWX denotes the interest level for the $X_{th}$ keyword. The order of X can be seen as the order of the posters which are visited by an agent. DX denotes the duration of KWX. The arrow-lines from one KW node to another KW node denotes the impact of the preceding keyword one the current one.

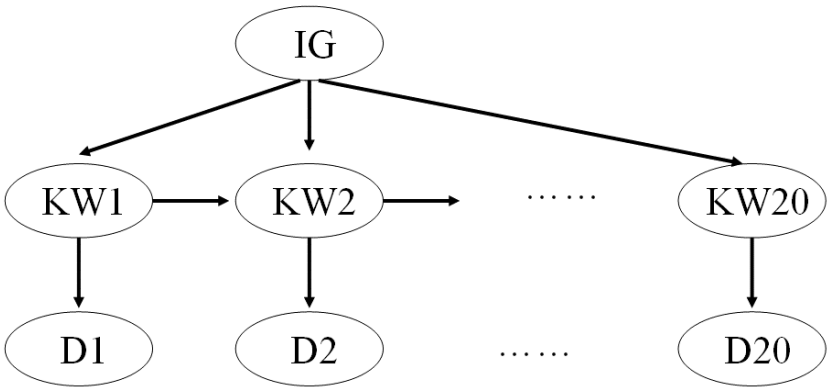



Figure 4.6: the whole Bayesian network

So far, we have designed the model of the Bayesian network and defined the states for every variable. However, we have not assigned the conditional probability table for them. If there is no conditional probability table, the Bayesian network is incomplete. We will report on how to compute the conditional probability table in Chapter 5.

### *4.4.2 Transfer the Network into the Junction Tree*

To illustrate the procedure of transferring the Bayesian network to the junction tree clearly, we have to assign condition probability tables for nodes. For now, we use an example set of condition probability tables. These conditional probabilities are generated by training the network using a set of artificial data. This set of artificial data is created by using the rules method described in Chapter 5. The conditional probability tables are shown in Table 4.3. We assign these probabilities into the Bayesian network. See Figure 4.7.

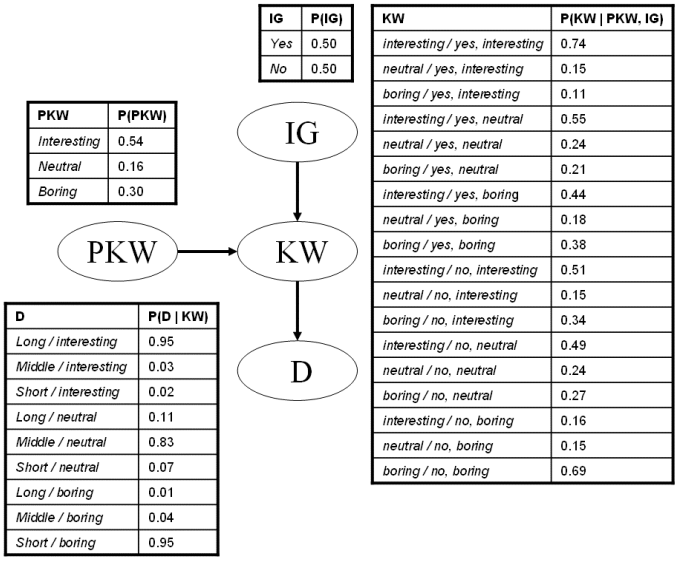
| P (IG) | |
|---|---|
| **State** | **Probability** |
| *yes* | 0.50 |
| *No* | 0.50 |

| P (PK) | |
|---|---|
| **State** | **Probability** |
| *interesting* | 0.54 |
| *neutral* | 0.16 |
| *boring* | 0.30 |

| P (D / K) | |
|---|---|
| **State** | **Probability** |
| *Long / interesting* | 0.95 |
| *Middle / interesting* | 0.03 |
| *Short / interesting* | 0.02 |
| *Long / neutral* | 0.11 |
| *Middle / neutral* | 0.83 |
| *Short / neutral* | 0.07 |
| *Long / boring* | 0.01 |
| *Middle / boring* | 0.04 |
| *Short / boring* | 0.95 |

| P (K / IG, PK) | |
|---|---|
| **State** | **Probability** |
| *interesting / yes, interesting* | 0.74 |
| *neutral / yes, interesting* | 0.15 |
| *boring / yes, interesting* | 0.11 |
| *interesting / yes, neutral* | 0.55 |
| *neutral / yes, neutral* | 0.24 |
| *boring / yes, neutral* | 0.21 |
| *interesting / yes, boring* | 0.44 |
| *neutral / yes, boring* | 0.18 |
| *boring / yes, boring* | 0.38 |
| *interesting / no, interesting* | 0.51 |
| *neutral / no, interesting* | 0.15 |
| *boring / no, interesting* | 0.34 |
| *interesting / no, neutral* | 0.49 |
| *neutral / no, neutral* | 0.24 |
| *boring / no, neutral* | 0.27 |
| *interesting / no, boring* | 0.16 |
| *neutral / no, boring* | 0.15 |
| *boring / no, boring* | 0.69 |

Table 4.3: the conditional probability tables




| IG | P(IG) |
|---|---|
| *Yes* | 0.50 |
| *No* | 0.50 |

| PKW | P(PKW) |
|---|---|
| *Interesting* | 0.54 |
| *Neutral* | 0.16 |
| *Boring* | 0.30 |

| KW | P(KW \| PKW, IG) |
|---|---|
| *interesting / yes, interesting* | 0.74 |
| *neutral / yes, interesting* | 0.15 |
| *boring / yes, interesting* | 0.11 |
| *interesting / yes, neutral* | 0.55 |
| *neutral / yes, neutral* | 0.24 |
| *boring / yes, neutral* | 0.21 |
| *interesting / yes, boring* | 0.44 |
| *neutral / yes, boring* | 0.18 |
| *boring / yes, boring* | 0.38 |
| *interesting / no, interesting* | 0.51 |
| *neutral / no, interesting* | 0.15 |
| *boring / no, interesting* | 0.34 |
| *interesting / no, neutral* | 0.49 |
| *neutral / no, neutral* | 0.24 |
| *boring / no, neutral* | 0.27 |
| *interesting / no, boring* | 0.16 |
| *neutral / no, boring* | 0.15 |
| *boring / no, boring* | 0.69 |

| D | P(D \| KW) |
|---|---|
| *Long / interesting* | 0.95 |
| *Middle / interesting* | 0.03 |
| *Short / interesting* | 0.02 |
| *Long / neutral* | 0.11 |
| *Middle / neutral* | 0.83 |
| *Short / neutral* | 0.07 |
| *Long / boring* | 0.01 |
| *Middle / boring* | 0.04 |
| *Short / boring* | 0.95 |

Figure 4.7: the Bayesian network

## Converting the DAG to a Junction Tree

According to the algorithm for constructing a junction tree, we can convert the DAG (the directed acyclic graph) into the junction tree as follows:

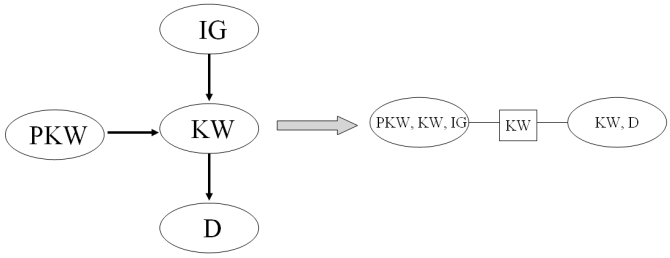



Figure 4.8: the junction tree corresponding to the DAG

Nodes “PKW, KW, IG” and “KW, D” are the clusters and “KW” is sepset which means separator nodes.

## Creating the Tables for Clusters and Sepset

For cluster “PKW, KW, IG”, the table is as follows:

| KW | IG | PKW | |
|---|---|---|---|
| i | y | i | 1 |
| n | y | i | 1 |
| b | y | i | 1 |
| i | y | n | 1 |
| n | y | n | 1 |
| b | y | n | 1 |
| i | y | b | 1 |
| n | y | b | 1 |
| b | y | b | 1 |
| i | n | i | 1 |
| n | n | i | 1 |
| b | n | i | 1 |
| i | n | n | 1 |
| n | n | n | 1 |
| b | n | n | 1 |
| i | n | b | 1 |
| n | n | b | 1 |
| b | n | b | 1 |

Table 4.4: initializing table for cluster “PKW, KW, IG”

Where “i”, “n” and “b” denote the states ”interesting”, ”neutral” and ”boring”, and “y” and “n” denote the states ”yes” and ”no”.

For cluster “KW, IG”, the table is as follows:

| D | KW | |
|---|---|---|
| l | i | 1 |
| m | i | 1 |
| s | i | 1 |
| l | n | 1 |
| m | n | 1 |
| s | n | 1 |
| l | b | 1 |
| m | b | 1 |
| s | b | 1 |

Table 4.5: initializing table for cluster “KW, D”

Where “l”, “m” and “s” denote the states “long”, “middle” and “short”.

For sepset “KW”, the table is as follow:

| KW | |
|---|---|
| i | 1 |
| n | 1 |
| b | 1 |

Table 4.6: initializing table for sepset “KW”

### Initializing the Tables

We initialize each cluster table ( $T_c$ ). For each variable $V_i$ , choose one cluster C containing $V_i$ and all its parents (there will be at least one such cluster). Let $T_c = T_c \times P(V_i \mid Parent(V_i))$ . (By $Parent(V_i)$ we mean in the original Belief Network.)

At the moment, we have two cluster tables as follows:

| KW | IG | PKW | P(IG)P(PKW)P(KW \| IG, PKW) |
|---|---|---|---|
| i | y | i | 0.50 x 0.54 x 0.74 = 0.1998 |
| n | y | i | 0.50 x 0.54 x 0.15 = 0.0405 |
| b | y | i | 0.50 x 0.54 x 0.11 = 0.0297 |
| i | y | n | 0.50 x 0.16 x 0.55 = 0.0440 |
| n | y | n | 0.50 x 0.16 x 0.24 = 0.0192 |
| b | y | n | 0.50 x 0.16 x 0.21 = 0.0168 |
| i | y | b | 0.50 x 0.30 x 0.44 = 0.0660 |
| n | y | b | 0.50 x 0.30 x 0.18 = 0.0270 |
| b | y | b | 0.50 x 0.30 x 0.38 = 0.0570 |
| i | n | i | 0.50 x 0.54 x 0.51 = 0.1377 |
| n | n | i | 0.50 x 0.54 x 0.15 = 0.0405 |
| b | n | i | 0.50 x 0.54 x 0.34 = 0.0918 |
| i | n | n | 0.50 x 0.16 x 0.49 = 0.0392 |
| n | n | n | 0.50 x 0.16 x 0.24 = 0.0192 |
| b | n | n | 0.50 x 0.16 x 0.27 = 0.0216 |
| i | n | b | 0.50 x 0.30 x 0.16 = 0.0240 |
| n | n | b | 0.50 x 0.30 x 0.15 = 0.0225 |
| b | n | b | 0.50 x 0.30 x 0.69 = 0.1035 |

Table 4.7: the CPT for cluster “PKW, KW, IG” assigning the probabilities

| D | KW | P(D \| KW) |
|---|---|---|
| l | i | 0.9500 |
| m | i | 0.0300 |
| s | i | 0.0200 |
| l | n | 0.1100 |
| m | n | 0.8300 |
| s | n | 0.0700 |
| l | b | 0.0100 |
| m | b | 0.0400 |
| s | b | 0.9500 |

Table 4.8: the CPT for cluster “KW, D” assigning the probabilities

The sepset table is still as follows:

| KW | |
|---|---|
| i | 1 |
| n | 1 |
| b | 1 |

Table 4.9: the CPT for sepset “KW”

### Making a Consistent Junction Tree

We will not repeat the message passing here, as the steps have been described in Chapter 2. The following tables are the CPTs (the conditional probability tables) after message passing. Through passing message, the consistency of the junction tree will be achieved.

| KW | IG | PKW | $\phi(KW, IG, PKW)$ |
|---|---|---|---|
| i | y | i | 0.20036937 |
| n | y | i | 0.04177267 |
| b | y | i | 0.03044383 |
| i | y | n | 0.04343345 |
| n | y | n | 0.01921406 |
| b | y | n | 0.01616645 |
| i | y | b | 0.06555526 |
| n | y | b | 0.02709643 |
| b | y | b | 0.05794824 |
| i | n | i | 0.13864495 |
| n | n | i | 0.04108378 |
| b | n | i | 0.09068536 |
| i | n | n | 0.03856914 |
| n | n | n | 0.01883977 |
| b | n | n | 0.02077712 |
| i | n | b | 0.02461582 |
| n | n | b | 0.02175485 |
| b | n | b | 0.10302939 |

Table 4.10: the CPT for cluster “PKW, KW, IG” after message passing

| D | KW | $\phi(KW, D)$ |
|---|---|---|
| l | i | 0.50402474 |
| m | i | 0.00710270 |
| s | i | 0.00006060 |
| l | n | 0.01842133 |
| m | n | 0.14005379 |
| s | n | 0.01128644 |
| l | b | 0.00003890 |
| m | b | 0.01558981 |
| s | b | 0.30342168 |

Table 4.11: the CPT for cluster “KW, D” after message passing

| KW | $\phi(KW)$ |
|---|---|
| i | 0.51118803 |
| n | 0.16976155 |
| b | 0.31905037 |

Table 4.12: the CPT for sepset “KW” after message passing

**Procedure of Inference**

So far, we have obtained a consistent junction tree. Therefore, we can use it to infer the probability of each node. Actually, we are interested in the node “KW”, because the probability of “KW” denotes how much interest the agent has in the keyword.

## 4.5 The Implementation of the Matchmaker

The matchmaking in our project is a sort of profile matchmaking. Profile matchmaking can be addressed by a variety of techniques, ranging from simple bipartite graph matching (with or without cost minimization), to vector-based techniques taken from classical Information Retrieval, to record matching in databases, among others.

How to present the user’s profile becomes a problem. We use a restriction of the ALC Description Logic described in Chapter 2. Apart from concepts and roles to represent the properties of (abstract) objects, it also allows one to express quantitative properties of objects, such as weight, length, by means of concrete domains. The next part will illustrate how to present the user’s file using the ALC Description Logic.

### *4.5.1 Presenting Users’ Profile*

We describe how to represent user profiles using the ALC Description Logic presented in Chapter 2. The user profiles are specifically tailored for our project. We do not use the full expressive power of the Description Logic. In particular, we use a single role *hasInterest*, to express interest in topics, and we make a limited use of the constructs. We assume the set of features to represent other characteristics such as name, company, and so on. Furthermore, we use a special feature level that expresses the interest level in a certain field. The concrete domain associated to interest level is the interval $\{\lambda \in \Re | 0 \leq \lambda \leq 1\}$ ($\Re$ is the real numbers)

A user profile P consists of the conjunction of the following parts:

- A conjunction of atomic concepts, which represents atomic properties related to the user. We denote the set of such concepts as Names (P).
- A conjunction of concepts of the form p (f), to represent physical characteristics. The unary predicate p can be one of the predicates $\geq \lambda(), \leq \lambda(), = \lambda()$ where $\lambda$ is a value of the concrete domain associated to f, or any logical conjunction of them. We denote the set of such concepts as Features (P). Since (p1 ^ p2)(f) is equivalent to p1(f) ∩ p2(f), in the following, we can assume that Features (P) contains at most one concept of the form p(f) for each feature f.
- A conjunction of concepts of the form $\exists hasInterest$ . $C\mathrm{I} \geq x(level)$, where C is a conjunction of concept names (keywords in our project), and $0 \leq x \leq 1$. Each such concept represents an interest in a concept C with level at least x. We denote the set of such concepts as $Interest$ (P).
- A conjunction of concepts of the form $\forall hasInterest$ . $C\mathrm{Y} \leq x(level)$, where C is a conjunction of concept names (keywords in our project), and $0 \leq x \leq 1$. Each such concept represents the fact that the interest in a concept C has level at most x. Note that, to represent the complete lack of interest in C, it is sufficient to put $x = 0$. We denote the set of such concepts as $NoInterest$ (P).

**Example 4.1:** Consider an agent, ID of 0101, with an email address yli89@gmail.com, the company is CSSE, and who has strong interests in novels and basketball, fair interest in movies and no interest in idols. This could be expressed as follows:

$$0101(ID) \cap = yli89@gmail.com(email) \cap = CSSE(company) \cap$$
$$\exists hasInterest(Novels \cap \geq 0.8(level)) \cap$$
$$\exists hasInterest(Basketball \cap \geq 0.8(level)) \cap$$
$$\exists hasInterest(Movies \cap \geq 0.4(level)) \cap$$
$$\forall hasInterest(Idols \cap \leq 0(level))$$

Where ID, email and company are the atomic concepts. $\exists hasInterest(Keyword \cap \geq Value(level))$ is the atomic role. In the example, the agent has two interests, novel which has a level over 0.8, and basketball with a level also over 0.8. He also likes watching movies but the level for this is not high, and he does not like idols. We suppose that interests are organized in a hierarchy

including $Novels \subseteq Books$ , and $Basketball \subseteq Sport$ . Therefore, we could say he most likes reading and sport.

Here, we explain why we have an interest hierarchy. For every pair of keywords K1 and K2, role R,

$$\text{If } \mathrm{H} \models K_1 \subseteq K_2 \text{ then } \mathrm{H} \models \exists R.(K_1 \mathrm{I} \geq_\lambda (f)) \subseteq \exists R.(K_2 \mathrm{I} \geq_\lambda (f))$$

For example, if there is $Basketball \subseteq Sport$ , then someone with a level of interest $\lambda$ in $Basketball$ has at least the same level of interest in $Sport$ .

### *4.5.2 Matchmaking Algorithm*

#### Description of the Algorithm

Although we have used the ALC Description Logic to represent the atomic concepts, such as Name (P), ID (1032) and so on, we do not use those atomic concepts in the matchmaking algorithm. The original algorithm [37] was created for a web dating service. We revise the original one so that it can be suitable for our project. To simplify the task, the algorithm can match two profiles which just contain the interest table, so matchmaking is performed according to the agent's interest table.

The matching is performed over two interest tables: the host table $Th$ and the neighbor table $Tn$ . The algorithm is symmetric, that is to say it evaluates how $Tn$ matches $Th$ , which is the same as how $Th$ matches $Tn$ . The algorithm has been divided into two parts: contraction and abduction. Contraction either removes or weakens interests from $Th$ so as to make $Th \cap Tn$ (conjunction) satisfiable in the hierarchy. Abduction, instead, either adds or strengthens interest in $Tn$ so as to make $Tn \in Th$ . The algorithm is based on structural algorithms for satisfiability and subsumption. Since it is reasonable to assume that users do not enter contradictory information, we assume that the profiles $Th$ and $Tn$ are consistent.

The result of the match is a penalty in $\Re$ : the larger the penalty, the less $Tn$ is suited for *Th*. In particular, partial penalties are added to the overall penalty by matching

corresponding conjuncts of the two profiles; this is done in two ways, by contraction and by abduction.

**Contraction**

When an interest $Ih$ in $Th$ conflicts with some interest $In$ in $Tn$, then $Ih$ is removed and a penalty is added. Intuitively, since the neighbor has something the host does not like, in order to make the profiles match the host gives up one of his/her requests. For example, let $Ih$ = (Sport, 0.2, -), and $In$ = (Basketball, 0.4, +), where we have Basketball is a child of sport in the hierarchy. In this case the host looks for someone who does not like sports very much, while the neighbor likes basketball and therefore he likes sports. In this case, pursuing the match would require the host to give up his or her request about sports, so the algorithm adds a penalty $Penalty\_CL(0.4,0.2)$ that depends on the gap between the lower bound (0.4) of the neighbor and the upper bound (0.2) of the host.

**Abduction**

When an interest $Ih$ in $Th$ has no corresponding interest in $Tn$, we add a suitable interest $In$ in $Tn$ that makes the profiles match, and add a corresponding penalty. Intuitively, the host wants something which the neighbor does not provide explicitly; in this case we assume that the neighbor may or may not satisfy the host's request, and as a consequence of this possibility of conflict we add a penalty. This is done by means of a penalty function $Penalty\_A()$, whose argument is an interest $I$, which takes into account the addition of $I$ to $Tn$. When the level of interest must be strengthened, we use a function $Penalty\_AL()$, which takes into account the gap between bounds.

**Algorithm:**

*CalculatePenalty*

**Input:** host interest table $Th$, neighbor interest table $Tn$, interest hierarchy $H$

**Output:** real value penalty > 0;

penalty = 0;

// **Contraction**

$(Ih, Xh, +) \in InterestTable(Th)$

**foreach** $(Ih, Xh, +) \in InterestTable(Th)$ do

    **foreach** $(In, Xn, -) \in InterestTable(Tn)$ do

**if** $Ih$ is a child of $In$ in $H$ and $Xh \geq Xn$

**then** replace $(Ih, Xh, +)$ in $Th$ with $(Ih, Xn, +)$

penalty = penalty + $Penalty\_CL(Xh, Xn)$

**foreach** $(Ih, Xh, -) \in InterestTable(Th)$ do

**foreach** $(In, Xn, +) \in InterestTable(Tn)$ do

**if** $In$ is a child of $Ih$ in $H$ and $Xh \leq Xn$

**then** replace $(Ih, Xh, -)$ in $Th$ with $(Ih, Xn, -)$

penalty = penalty + $Penalty\_CL(Xn, Xh)$

// **Abduction**

**foreach** $(Ih, Xh, +) \in InterestTable(Th)$ do

**if** there does not exist $(In, Xn, +) \in InterestTable(Tn)$ such that $In$ is a child of $Ih$ in $H$ and $Xn > Xh$

**then if** there exists $(In, Xn, +) \in InterestTable(Tn)$ such that $In$ is a child of $Ih$ in $H$

**then** let $(In, Xn, +)$ be the interest in interest table $Tn$ with maximum $X$ among those for which $In$ and $Ih$ hold penalty = penalty + $Penalty\_AL(Xh, Xn)$

**else** penalty = penalty + $Penalty\_A(In, Xh, +)$ add $(Ih, Xh, +)$ to $Tn$

**foreach** $(Ih, Xh, -) \in InterestTable(Th)$ do

**if** there does not exist $(In, Xn, -) \in InterestTable(Tn)$ such that $Ih$ is a child of $In$ in $H$ and $Xh > Xn$

**then if** there exists $(In, Xn, -) \in InterestTable(Tn)$ such that $Ih$ is a child of $In$ in $H$

**then** let $(In, Xn, -)$ be the interest in interest table $Tn$ with minimum $X$ among those for which $In$ and $Ih$ holds penalty = penalty + $Penalty\_AL(Xn, Xh)$

**else** penalty = penalty + $Penalty\_A(In, Xh, -)$ add $(Ih, Xh, -)$ to $Tn$

return penalty

**Penalty Function:**

**Float** $Penalty\_CL(a,b)$

{

**If** $a,b \in [0,1]$

**Then** return $a-b$.

}

**Float** $Penalty\_AL(a,b)$

{

**If** $a,b \in [0,1]$

**Then** return $\frac{a-b}{1-b}$.

}

**Float** $Penalty\_A(1,X,+)$

{

**If** $X \in [0,1]$

**Then** return $X$.

}

**Float** $Penalty\_A(1,X,-)$

{

**If** $X \in [0,1]$

**Then** return $1-X$.

}

This algorithm allows the profile description to be incomplete in the parts that are unavailable or are considered irrelevant by the user. Moreover, it uses the property of interest hierarchy so that it can perform more accurately. Finally, it can be extended according to different requirements. For example, we calculate the similarity based on not only the interest, but also the user's name, features, even job.

## 4.6 Summary

In this chapter, we have illustrated the procedure for implementing the inference model and the matchmaker.

- The requirements for implementing the inference model and the matchmaker.
- The architecture of the server system.
- The implementing procedure of the inference module.
- The implementing procedure of the matchmaker.

Chapter 5

# THE EVALUATION PLAN

After implementing the Bayesian network, several important questions come up. Can the network really infer the people's interests using limited data? How accurate is it? Can it be suitable for any environment? To explore these questions we conducted a set of experiments.

Our hypothesis is that the network will accurately infer people's interests, and will also perform very well in different environments. This chapter presents the evaluation plan used in experiments that test this hypothesis. Artificial data will play an important role in the experiments, as discussed in the next section. The universality problem will be caused by the use of the artificial data, which means whether or not the network trained by one sort of the artificial data has the capability to fit other sorts of data, such as the data collected from the real environment. The solution will be shown in section 5.2, where we also present a plan to evaluate the Bayesian network using the artificial data, and the plan of the evaluation which will combine the simulation application and the inference system.

## 5.1 The Artificial Data

### *5.1.1 What is the artificial data?*

For training and testing the Bayesian network, we have to use a data set. Generally, the data set is collected from a number of real experiments. It can provide the evidence that the network is able to figure out the probability of every node. On the other hand, if the data set cannot be collected from experiments, but is instead generated by some artificial methods, then we call it the artificial data. The artificial data can also provide evidence to train the network and test the network. In our evaluation we use artificial data to train and test the network.

### 5.1.2 Why do we use the artificial data?

Artificial data was used because we did not have the time to organize a real conference, or the resources to create a large number of working smart badges for the conference.

### 5.1.3 What problems does the artificial data cause?

There is a problem with using artificial data. If we use an artificial data set to train the network, and then also test the network using another artificial data set, these artificial data sets cannot be generated by the same method. If they are a universality problem will arise. This means we cannot say the network is general to other data set generated by other ways. Therefore, we need to create two sets of artificial data using two different methods. In the following section, we will illustrate how these two different sets of data solve this problem.

### 5.1.4 What does the artificial data represent?

We suppose that the agents would watch the posters from Poster1 to Poster20 in order. Each poster he is looking at contains one interest keyword. We may generate a group of artificial data for each agent. The group of data includes the person ID and 20 sets of state lists which represents what state the nodes of the network are in, and when the agent watches the poster. See the following sample (Figure 5.1).

Person ID: COSC000

Keyword ID: KW0
'Interest Group': No
'the Last Keyword': Interesting
'Keyword': Interesting
'Duration': Long
Keyword ID: KW1
'Interest Group': Yes
'the Last Keyword': Interesting
'Keyword': Interesting
'Duration': Long
……
Keyword ID: KW19
'Interest Group': Yes
'the Last Keyword': Boring
'Keyword': Boring
'Duration': Short

Figure 5.1: the example of the Yi's artificial data set

This example is showing a part of Yi's artificial data set. Firstly, it contains the person ID which is COSC000. Next is the state list which contains the state of every node of the network for 20 posters. For example, the first keyword ID is KW0. When Yi watches the poster which contains KW0, the state of "Interest Group" is NO, the state of "the Last Keyword" is INTERESTING, the state of "Keyword" is INTERESTING and the state of "Duration" is LONG. The network will be trained and tested by this kind of artificial data.

## 5.2 The Evaluation Plan

The evaluation has three important purposes. One is to evaluate the accuracy of the network inferring the interests. The accuracy of the inference means how many of the interest results worked out by the network are correct. We will compare the results with the original interest record, and figure out the accuracy. The second purpose is to evaluate whether the network is general, i.e. that the network does not have the universality problem. The last one is to use the simulation application to simulate

various circumstances so that the inference system can show it is able to perform very well under these conditions.

The solution of the universality problem is to design two methods which are able to generate two different sets of artificial data. The two data sets have entirely different content. We will apply rules and probabilistic methods to generate the two data sets. The first method is to use the constrained rules, such as if the interest level of the agent for a keyword is over 5, then he must stop at front of the poster which contains the keyword for over 4 minutes. The second way is to use a probabilistic approach, for example, if the interest level of the agent for a keyword is 5, then he will have to stay at front of the poster which contains the keyword over a piece of duration. The duration is determined by the probability, 20% is 1 minute, 10% is 3 minute, 70% is over 4 minutes.

To prove the universality of the network, we first train the network using the artificial data from rules. After that, the network will be tested by the data from the probabilistic approach. Since the ways to create the data in each of these approaches is different, the levels of random of the data creation are also different. Secondly, we swap the roles of the data. We train the network using the data from rules, and then test it using the data from the probabilistic approach. If the inference accuracy of the network remains high, we can say the network is general. Meanwhile, since we have used different data to train and test the network, we can say the universality problem has been solved. Section 5.2.1 will cover what are the requirements for designing rules and probabilistic way. After that, the rest of Section 5.2 will represent the plan to achieve the three purposes.

### *5.2.1 The Work before Evaluation*

#### Step1: Designing a Model for Each Person

Before evaluation, we have to design a model for each artificial person. The model includes an ID, the name and the keyword table. There will be 50 people, each with an exclusive ID number and the name. Furthermore, each person will have a keyword

table. The keyword table will contain the keyword ID and the interest level corresponding to the 20 keywords.

Those models are the base of the artificial data, because the artificial data will be created by some methods according to these models. Meanwhile, in the simulator, we also use those models to create the agents' behavior when we evaluate the network and the matchmaker.

### Step 2: Developing a Set of "Rules"

The artificial data mainly represents the state list of the nodes of the network. Due to the models for each person, we can easily figure out the value of "Interest Group", "Preceding Keyword" and "Keyword". The rules are in charge of generating the state of "Duration" according to the above three values. The rules should be reasonable and the habits of the people should be considered. For example, when a person has strong interest in a certain object, then he will spend a longer time looking at this object. More detail about this will be described in the next chapter.

### Step 3: Developing the Probabilistic Way

The requirements for this step are almost same with the last step. The only difference is that we have to design the probabilities instead of the rules. We can obtain the probabilities from our own experience or other reference. For example, there is an inertia when people are looking at posters one by one. See Table 5.1. P(KW/PKW) means the conditional probability of the current keyword. When the prior keyword is interesting, the probability of the current keyword is also interesting is 0.6, the probability of the current keyword is boring is 0.2. This table presents a human inertia when a person just finished the prior poster, he is likely to think the next poster would be similar with the last he just watched.

| P(KW/PKW) | INTERESTING | NEUTRAL | BORING |
|---|---|---|---|
| INTERESTING | 0.60 | 0.20 | 0.20 |
| NEUTRAL | 0.20 | 0.60 | 0.20 |
| BORING | 0.20 | 0.20 | 0.60 |

Table 5.1: the inertial probability

### *5.2.2 Evaluation One: Does it Work Well and Is It General?*

The purpose of the first evaluation is to verify whether or not the Bayesian network is accurate. To achieve that, different combinations of the artificial data sets will be used so that we can acquire a more accurate result. We divide the first evaluation into 5 steps as following:

1. Use the model of each agent to create a set of artificial data for each agent using the rules, giving the duration the agent spends at each poster.
2. Train the network using the artificial data which is created in the first step.
3. Test the network using the SAME artificial data and compare the resulting personal models learned for each artificial agent to the personal models used to generate the data. Are they the same? Does it correctly predict from the data what interests the agent has? This tests that the network is capturing all the required dependencies.
4. Create another different set of artificial data for each agent using the rules again.
5. Retest the resulting models using data created in step 4. Does this data correctly infer what the agent's interest is?

### *5.2.3 Evaluation Two: Is It General?*

The evaluation will evaluate whether or not the network fits other data which has been created using a different approach. Therefore, we will repeat the steps of the first Evaluation, but the artificial data will be generated using the probabilistic method.

1. For the personal models learned in the first test, test them using the data created for the second test. The Bayesian network is trained on non-probabilistic data, but tested using probabilistic data, and see how well it does, for a variety of parameters

### *5.2.4 Evaluation Three: Are there other suitable circumstances?*

To achieve this point, we design a circumstance which simulates a conference environment. This circumstance will be simulated by the simulation application. The evaluation has four goals:

1. How robust is the network at predicting interests under simulated circumstances?
2. How robust is the matchmaker at calculating the similarity of the simulated agents?
3. How does the network affect the agent's behavior?
4. How does the agent's behavior affect the ability of the network?

The first goal is expected to prove that the inference network could capture the necessary data to perform the prediction. The second goal is to verify the matchmaker using the simulator. The third and fourth goals are to analyze the interaction between the network and the agent's behavior according to the data we collect

The simulator simulates a conference. The attendees will be permitted to talk to each other and look at posters in the virtual conference. The simulator will generate the duration the agent spends on each poster, and how long the agent spends in conversation. The inference network will predict the interests of each person through the data received from the simulation. The matchmaker will calculate the similarity of the agents in real time.

**Circumstances:**

The agents are allowed to look at the posters and talk to each other.

**Steps:**

**1.** The simulator presents a virtual conference. The agents will be allowed to look at the posters and talk to each other. Meanwhile the duration spent at each poster will be recorded. The time spent in conversation will also be recorded. Finally the network will work out the interests of the people.

**2.** Every agent's interest table will be figured out.

3. The accuracy will be worked out according to comparison of the new interest tables and the original one of the personal models.
4. According to the original personal models, we calculate all the similarity values among the 50 agents. We call those the original similarities. We figure out the relationship states among the 50 agents based on the original similarities. The relationship state has three values which are FRIENDLY, NEUTRAL and NOT FRIENDLY.
5. We will also calculate all similarity values among the 50 agents first according to the new interest tables which are created in the second step. Since the new interest table will be inferred in real time, the similarity among the 50 agents should also be revised in real time. We consider the last revised result as the final result.
6. According to the similarity, we also figure out the relationship states among the 50 agents.
7. We will compare the relationship distribution to the original one.
8. We will deeply analyze the interaction between the network and the agent's behavior.

## 5.3 Summary

This chapter presents a plan for the evaluation of the server system. We will conduct three evaluations. The first one is to verify whether the network is able to work or not. The second one is to prove the network is general. Finally, the last one is to generate a complex circumstance using the simulator and evaluate the network and the matchmaker in the same time.

Chapter 6

# THE EVALUATION

In this chapter we report on the evaluation result. In the first section, we present the preparation work performed before the evaluations. In terms of the evaluation plan, we needed to perform three tests. In the following sections, we respectively introduce the procedure of the evaluations and the results in detail. In each section, we will make a conclusion and summarize the evaluation. In the end of the chapter, we will discuss what we learnt from the evaluation, what are the differences between a real world and simulated environment, and present the assumptions which can cause the network to be invalid in real world.

## 6.1 The Preparation Work

In this section, we report on how to create the personal model for each person, and then introduce the rules and the probabilistic method we used to be designed the models.

### *6.1.1 Design a Model for Each Person*

In our evaluation we use fifty virtual agents, although generally speaking, this number could be much higher. Each agent has a personal model which includes the person's ID, name, and an interest table. In the interest table, we assigned every keyword one keyword ID number, for example the ID of 'BASKETBALL' is 2. Meanwhile, we also assigned a real number to the interest's levels corresponding to each keyword. The range of the real number was from 0.0 to 1.0. There were 20 keywords in each interest table. The keywords and keyword's ID's corresponding relationship are shown in Table 4.1 in Chapter 4.

| **ID** | COSC01 | |
|---|---|---|
| **Name** | Yi Liu | |
| **The Keyword Table** | **ID** | **Level** |
| | 1 | 0.8 |
| | 2 | 0.7 |
| | 3 | 0.9 |
| | 4 | 0.6 |
| | 5 | 0.5 |
| | 6 | 0.5 |
| | 7 | 0.4 |
| | 8 | 0.3 |
| | 9 | 0.3 |
| | 10 | 0.4 |
| | 11 | 0.2 |
| | 12 | 0.3 |
| | 13 | 0.8 |
| | 14 | 0.9 |
| | 15 | 0.7 |
| | 16 | 0.8 |
| | 17 | 0.3 |
| | 18 | 0.4 |
| | 19 | 0.3 |
| | 20 | 0.2 |

Table 6.1: an example of the Interest Model for Yi Liu

There is an example (Table 6.1) model for ‘Yi Liu’. As you can see, in the keyword table, the interest’s levels were generated arbitrarily. A special generator was created for generating this data, and we generated 50 models that could be used by other programs. The overall set of 50 models used is shown in Appendix A.

### *6.1.2 Develop a Set of "rules"*

Rules were used to generate the amount of time a person spent looking at a poster with a particular keyword. Figure 6.1 shows the rules used.

```
If 'KW' = Interesting, then Time = (7 - 9) minutes
If 'KW' = Neutral, then Time = (4 - 6) minutes
If 'KW' = Boring, then Time = (0 - 3) minutes

If IsOptional () = TRUE, then
{
        If 'PKW' = Interesting, then Time = Time + 2 minutes
        If 'PKW' = Neutral, then Time = Time + 0 minutes
        If 'PKW' = Boring, then Time = Time - 2  minutes

        If 'IG' = Yes, then Time = Time + 2 minutes
        If 'IG' = No, then Time = Time + 0 minutes
}

If Time > 7 then 'D' = Long
If 3> Time > 7 then 'D' = Middle
If 0 > Time > 3 then 'D' = Short

Note:
'KW' is 'Keyword'
'PKW' is 'Preceding Keyword'
'IG' is 'Interest Group'
'D' is 'Duration'
```

Figure 6.1: the rules generating the artificial data

Since the duration is mainly based on the interest level of the person looking at the poster (keyword), the rules determine the basic durative time based on the value of 'KW'. The function 'IsOptional ()' is used to make the decision about whether the 'PKW' and 'IG' have the capability to affect the durative time generated in the first step. The reason why the effect is optional is that the prior keyword and the state of the interest group cannot change the duration of looking at the poster at all the time. Therefore, 'IsOptional ()' makes this decision randomly. In short, the rules have to create a reasonable duration.

Take the Yi model for example, in terms of Table 6.1 when Yi is watching poster KW19, the 'IG' is Yes, the 'PKW' is 'Boring' and the 'KW' is 'Boring'. The states of 'IG' are YES and NO; the states of 'PKW' and 'KW' are INTERESTING, NEUTRAL and

BORING; the states of ‘D’ are LONG, MIDDLE and SHORT. Consequently, the basic durative time is figured out by the rules, with a result of 0 – 3 minutes. If ‘IsOptional ()’ is true, then the durative time would still be 0 – 3 minutes probably since ‘IG’ is Yes and ‘PKW’ is ‘Boring’. Finally, the last result could be the value of Short.

### *6.1.3 Develop the Probabilistic Way*

We also used a probabilistic method to create artificial data with the same format. The following table shows the probabilities we used.

| KW | PKW | IG | | P (Long) | P (Middle) | P (Short) |
|---|---|---|---|---|---|---|
| interesting | interesting | yes | | 0.90 | 0.05 | 0.05 |
| interesting | interesting | no | | 0.85 | 0.10 | 0.05 |
| interesting | neutral | yes | | 0.85 | 0.10 | 0.05 |
| interesting | neutral | no | | 0.75 | 0.15 | 0.05 |
| interesting | boring | yes | | 0.75 | 0.20 | 0.05 |
| interesting | boring | no | | 0.70 | 0.20 | 0.10 |
| neutral | interesting | yes | | 0.05 | 0.90 | 0.05 |
| neutral | interesting | no | | 0.05 | 0.85 | 0.10 |
| neutral | neutral | yes | | 0.10 | 0.80 | 0.10 |
| neutral | neutral | no | | 0.05 | 0.75 | 0.20 |
| neutral | boring | Yes | | 0.10 | 0.70 | 0.20 |
| neutral | boring | No | | 0.10 | 0.60 | 0.30 |
| boring | interesting | Yes | | 0.10 | 0.25 | 0.75 |
| boring | interesting | No | | 0.10 | 0.15 | 0.85 |
| boring | neutral | Yes | | 0.10 | 0.20 | 0.70 |
| boring | neutral | No | | 0.10 | 0.15 | 0.80 |
| boring | boring | Yes | | 0.05 | 0.10 | 0.85 |
| boring | boring | No | | 0.05 | 0.05 | 0.95 |

Table 6.2: the probability table for the probabilistic way

In Table 6.2, we can see there are 3 columns on the left hand, which are presenting the states for the nodes ’KW‘, ’PKW‘ and ’IG‘. ’KW‘ denotes the node of “Keyword”. ’PKW‘ is the abbreviation of the node of “the Prior Keyword”. ’IG‘ means the node of “the Interest Group”. Every row of this table covers one state combination of the three nodes, and the probabilities of the node of “Duration” are shown at the right hand corresponding to every state combination. For example, the first row, we say when the “KW” is INTERESTING, “PKW” is INTERESTING and “IG” is YES, the probabilities for “Duration” can be P(LONG) = 0.90, P(MIDDLE) = 0.05 and P(SHORT) = 0.05. Furthermore, the probability values are determined by that if a person is interested in a certain object, then he probably spends a longer time on it. [9]

## 6.2 Evaluation One: Does it Work Well?

The purpose of the first evaluation is to verify whether or not the Bayesian network is accurate. To achieve that, the different combinations of the artificial data sets are used so that we can acquire a more accurate result. We used the following 5 steps described in the last chapter:

1. It is to create the artificial data using rules.
2. It is to train the network using the data created in the last step.
3. It is to test the network using the same data set.
4. It is to create another different artificial data set using the rules.
5. It is to retest the network using the data set created in the last step.

For this purpose, we programmed an application called *DataGenerator* to generate the virtual agent data set. We applied the random algorithm to implement it. Therefore, the results each time were different, and also unpredictable. This point could guarantee the generality of the data set. This application could generate the artificial data using rules or a probabilistic method.

### *6.2.1 Creating the Artificial Data Using Rules*

Firstly, we needed to generate the training data set for training the network. We used a DataGenerator application to create the training data. At this time, we created the data set using rules. Figure 6.2 shows the resultant file.

As you can see, the first number 50 is the number of the agents followed by the first agent's ID. After that, the state list is shown. There are 20 groups corresponding to 20 keywords respectively. Each group has 4 states which are corresponding to 4 nodes of the network respectively. The first state is "IG", the second is "PKW", the third is "D", the forth is "KW". The second agent's ID which is 1 follows the state list. In the file there are 50 agents' information.

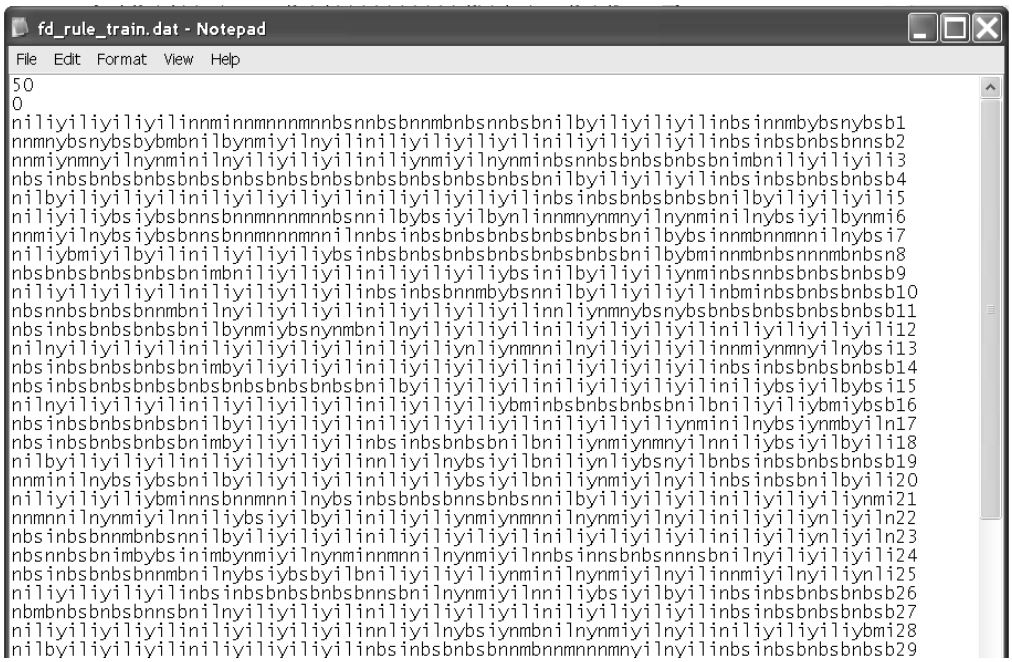



Figure 6.2: fd_rule_train.dat

In preparing for the evaluation we designed the personal interest model for each agent. Therefore, we knew the virtual avatar's interest level for each keyword so that we could figure out the state of the node "KW" according to the level. For example, take Agent 0. We were calculating his four states of the first keyword. Because of the first keyword, the state of "IG" should be NO, and we randomly assigned the state of "PKW" as INTERESTING. We could get the state of "KW" which was INTERESTING. Finally, we used the rules to figure out the state of "D" which is LONG. When we calculated the four states of the second keyword, the state of "PKW" was the state of the last keyword which was INTERESTING. We used this method to create a data set for 50 agents.

### *6.2.2 Training the Network Using the Above Data Set*

The learning problem can be summarized as follows:

| Structure | Observability | Method |
|---|---|---|
| Known | full | Sample statistics |
| Known | partial | EM or gradient ascent |
| Unknown | full | Search through model space |
| Unknown | partial | Structural Expected Model |

Table 6.3: the learning problem of Bayesian network

Full observability means that the values of all variables are known; partial observability means that we do not know the values of some of the variables. Unknown structure means we do not know the complete topology of the graph. Typically we will know some parts of it, or at least know some properties the graph is likely to have.

As for our case, the learning problem is known with full observability. We have already designed the structure of the network, and created complete artificial data for training. Therefore, in this case, the goal of learning is to find the values of the parameters of each CPD which maximizes the likelihood of the training data. Before that, we have to find the prior probability.

### Prior Probability

Firstly, we assigned two equal values to P(IG), because we do not have the prior knowledge for this. Secondly, we calculated out the P(D/KW) according to the experiment done in [38], in which the researcher performed a statistic about how long the visitors of a museum spend on an object based on the different interest and the different person styles. Finally, we designed the third table according to the inertia probability mentioned in Chapter 2 and the experiment. Thus the prior probability is:

For P(INTEREST GROUP)

| INTEREST GROUP | PROBABILITY |
|---|---|
| YES | 0.50 |
| NO | 0.50 |

Table 6.4: the prior for P(IG)

For P(DURATION / KEYWORD)

| KEYWORD | DURATION | PROBABILITY |
|---|---|---|
| INTERESTING | LONG | 0.60 |
| INTERESTING | MIDDLE | 0.32 |
| INTERESTING | SHORT | 0.08 |
| NEUTRAL | LONG | 0.43 |
| NEUTRAL | MIDDLE | 0.33 |
| NEUTRAL | SHORT | 0.24 |
| BORING | LONG | 0.28 |
| BORING | MIDDLE | 0.32 |
| BORING | SHORT | 0.40 |

Table 6.5: the prior for P(DURATION / KEYWORD)

For P(KEYWORD / INTEREST GROUP, PRIOR KEYWORD)

| PRIOR KEYWORD | INTEREST GROUP | KEYWORD | PROBABILITY |
|---|---|---|---|
| INTERESTING | YES | INTERESTING | 0.85 |
| INTERESTING | YES | NEUTRAL | 0.10 |
| INTERESTING | YES | BORING | 0.05 |
| INTERESTING | NO | INTERESTING | 0.75 |
| INTERESTING | NO | NEUTRAL | 0.15 |
| INTERESTING | NO | BORING | 0.10 |
| NEUTRAL | YES | INTERESTING | 0.50 |
| NEUTRAL | YES | NEUTRAL | 0.30 |
| NEUTRAL | YES | BORING | 0.20 |
| NEUTRAL | NO | INTERESTING | 0.30 |
| NEUTRAL | NO | NEUTRAL | 0.50 |
| NEUTRAL | NO | BORING | 0.20 |
| BORING | YES | INTERESTING | 0.25 |
| BORING | YES | NEUTRAL | 0.35 |
| BORING | YES | BORING | 0.40 |
| BORING | NO | INTERESTING | 0.20 |
| BORING | NO | NEUTRAL | 0.30 |
| BORING | NO | BORING | 0.50 |

Table 6.6: the prior probability for P(KEYWORD / PRIOR KW, INTEREST GROUP)

Table 6.4 shows the prior probability of the P(IG). IG denotes the node of the situation of the agent's interest group. It has two states, YES and NO. Therefore, I assigned the

prior probability as 0.5 for each state. Table 6.5 covers the prior probability of P(DURATION / KEYWORD). Table 6.6 covers the prior probability of P(KEYWORD / PRIRO KW, IG). These two resultant priors are from [38].

### Parameter Estimation

Since the nodes of the network were discrete, we could compute the Maximum Likelihood (ML) estimates by simple counting. For example, in the network, we had

$$P(K = I / IG = Y, PK = I) = \frac{N(K = I / IG = Y, PK = I)}{N(IG = Y, PK = I)}$$

Where $N(K = I / IG = Y, PK = I)$ was the number of times this event occurs in the training data set.

The resultant conditional probability distributions (CPDs) are as follows:

| **Node IG** | **P(IG)** | **Number** |
|---|---|---|
| YES | 0.50 | 502 |
| NO | 0.50 | 498 |
| **Sum** | | 1000 |
| | | |
| **Node PKW** | **P(PKW)** | **Number** |
| INTEREST | 0.54 | 543 |
| NEUTRAL | 0.16 | 157 |
| BORING | 0.30 | 300 |
| **Sum** | | 1000 |
| | | |
| **Node D** | **P(D / KW)** | **Number** |
| LONG/INTEREST | 0.98 | 492 |
| MIDDLE/INTEREST | 0.01 | 7 |
| SHORT/INTEREST | 0.01 | 7 |
| **Sum** | | 506 |
| LONG/NEUTRAL | 0.11 | 18 |
| MIDDLE/NEUTRAL | 0.83 | 137 |
| SHORT/NEUTRAL | 0.07 | 11 |
| **Sum** | | 166 |
| LONG/BORING | 0.05 | 16 |
| MIDDLE/BORING | 0.05 | 16 |
| SHORT/BORING | 0.90 | 296 |
| **Sum** | | 328 |
| | | |
| **Node K** | **P(KW / IG, PKW)** | **Number** |
| INTEREST/YES, INTEREST | 0.74 | 283 |
| NEUTRAL/YES, INTEREST | 0.15 | 59 |
| BORING/YES, INTEREST | 0.11 | 43 |
| **Sum** | | 385 |
| INTEREST/YES, NEUTRAL | 0.55 | 43 |
| NEUTRAL/YES, NEUTRAL | 0.24 | 19 |
| BORING/YES, NEUTRAL | 0.21 | 16 |
| **Sum** | | 78 |
| INTEREST/YES, BORING | 0.44 | 17 |
| NEUTRAL/YES, BORING | 0.18 | 7 |
| BORING/YES, BORING | 0.38 | 15 |
| **Sum** | | 39 |
| INTEREST/NO, INTEREST | 0.51 | 81 |
| NEUTRAL/NO, INTEREST | 0.15 | 24 |
| BORING/NO, INTEREST | 0.34 | 53 |
| **Sum** | | 158 |
| INTEREST/NO, NEUTRAL | 0.49 | 39 |
| NEUTRAL/NO, NEUTRAL | 0.24 | 19 |
| BORING/NO, NEUTRAL | 0.27 | 21 |
| **Sum** | | 79 |
| INTEREST/NO, BORING | 0.16 | 43 |
| NEUTRAL/NO, BORING | 0.15 | 38 |
| BORING/NO, BORING | 0.69 | 180 |
| **Sum** | | 261 |

Table 6.7: the CPDs of the network

In the above tables, the first column covers the events of the nodes; the second column shows the conditional probability of the nodes; the third column represents the number of times the particular event occurs. For example, in the CPD of node KW, the first three rows denote when the conditions are IG = YES and PKW = INTERESTING, the KEYWORD has 74% probability of being INTERESTING because there are 283 times the KEYWORD is INTERESTING in 385 which denotes the sum times of the IG = YES and PKW = INTERESTING.

### *6.2.3 Testing the Network using SAME Data Set*

So far, we have created a complete Bayesian network which contains a known structure and CPD for each node. In this section we present results from testing the network.

We are interested in the resultant state of the node K from the inference system when we use the training data set. However, the data set is incomplete in that is does not contain the state of KEYWORD.

#### Initiation

Since we had transferred the Bayesian network into the Junction tree, we initiate the junction tree with the CPDs created in the last step. According to the method mentioned in Chapter 2, we get a probability distribution table (PDT) for each the junction tree's node as follows:

| KW | PDT |
|---|---|
| Interesting | 1.00000000 |
| Neutral | 1.00000000 |
| Boring | 1.00000000 |

Table 6.8: the PDT for node K

| D | KW | PDT |
|---|---|---|
| Long | Interesting | 0.98598695 |
| Middle | Interesting | 0.01389449 |
| Short | Interesting | 0.00011855 |
| Long | Neutral | 0.10851295 |
| Middle | Neutral | 0.82500303 |
| Short | Neutral | 0.06648405 |
| Long | Boring | 0.00012191 |
| Middle | Boring | 0.04886315 |
| Short | Boring | 0.95101494 |

Table 6.9: the PDT for node D-KW

| KW | IG | PKW | PDT |
|---|---|---|---|
| Interesting | Yes | Interesting | 0.20034935 |
| Neutral | Yes | Interesting | 0.04176849 |
| Boring | Yes | Interesting | 0.03044079 |
| Interesting | Yes | Neutral | 0.04342911 |
| Neutral | Yes | Neutral | 0.01921214 |
| Boring | Yes | Neutral | 0.01616483 |
| Interesting | Yes | Boring | 0.06554871 |
| Neutral | Yes | Boring | 0.02709373 |
| Boring | Yes | Boring | 0.05794244 |
| Interesting | No | Interesting | 0.13863109 |
| Neutral | No | Interesting | 0.04107968 |
| Boring | No | Interesting | 0.09067629 |
| Interesting | No | Neutral | 0.03856529 |
| Neutral | No | Neutral | 0.01883789 |
| Boring | No | Neutral | 0.02077504 |
| Interesting | No | Boring | 0.02461336 |
| Neutral | No | Boring | 0.02175267 |
| Boring | No | Boring | 0.10301910 |

Table 6.10: the PDT for node KW-IG-PKW

### The Result of Testing

After making the junction tree consistent, we can start to infer the state of the node we are interested in. The inputs of the inference system were the state of the network node IG, PKW and D. The output was the state of KEYWORD which was figured out according to the states of other nodes and their PDT. Since there were 50 agents and 20 posters, the number of times the system makes inferences was 50*20 = 1000. After that, we could get the inferred state of KEYWORD, and compared it to the

corresponding state in the model used to generate the data. We found the following results:

| | |
|---|---|
| The number of correct inferred state | 948 |
| The number of incorrect inferred state | 52 |
| The correct result ratio | 0.948 |

Table 6.11: the ratio of the correct result

From Table 6.11, we can see the accuracy of the network is 0.948. This means the network was able to predict from the data with almost 95% accuracy what interests the agent has.

### *6.2.4 Creating another Different Set of Artificial Data Using the Rules*

To prove the result was not coincidence, we tested with a different artificial data set. The second data set was also generated from the personal model. Although the interest level for each keyword was same with the training data, the duration values would be different because of the rules.

### *6.2.5 Retesting the Network Using Data Created in Last Step*

We retested the network using the data set created in last step Doing this we found the following results:

| | |
|---|---|
| The number of correct inferred state | 954 |
| The number of incorrect inferred state | 46 |
| The correct result ratio | 0.954 |

Table 6.12: the ratio of the correct result

Thus in this case we were also around 95% accurate in predicting the agent interest level. From the above table, we can draw a conclusion that the network has the capability to correctly infer what the agent's interest is.

### 6.2.6 Summary

The goal of this first evaluation is to prove the network can work, capture the correct data and predict the result with a high level of accuracy. In this evaluation, we first used a set of artificial data to train the network, and then used the same set to test it. The accuracy ratio was 0.948. We also generated another data set using the rules. We called this data set the testing data set as it had different content than the training set. We tested the network using the testing set, and the accurate ratio was 0.954. So the network was able to capture the correct data and perform well. The network cannot perform a 100% correct result because of the noise. For example, if an agent's prior keyword is interesting, the duration is middle and the state of Interest group is yes, the current keyword should be interesting according to the computation of the network. However, there is a special case in the testing data, in which current keyword is neutral. This can cause an incorrect result.

## 6.3 Evaluation Two: Is It General?

A second evaluation was conducted to evaluate whether or not the network fits other data created using different methods. We repeated the steps of the first evaluation, but the artificial data was generated using a probabilistic approach. Moreover, we would add another two steps to complete the evaluation.

### 6.3.1 Creating the Artificial Data Using the Probabilistic Way

We still used the DataGenerator program to generate the artificial data set in a probabilistic manner and store in the same file format as used previously (see Figure 6.3).

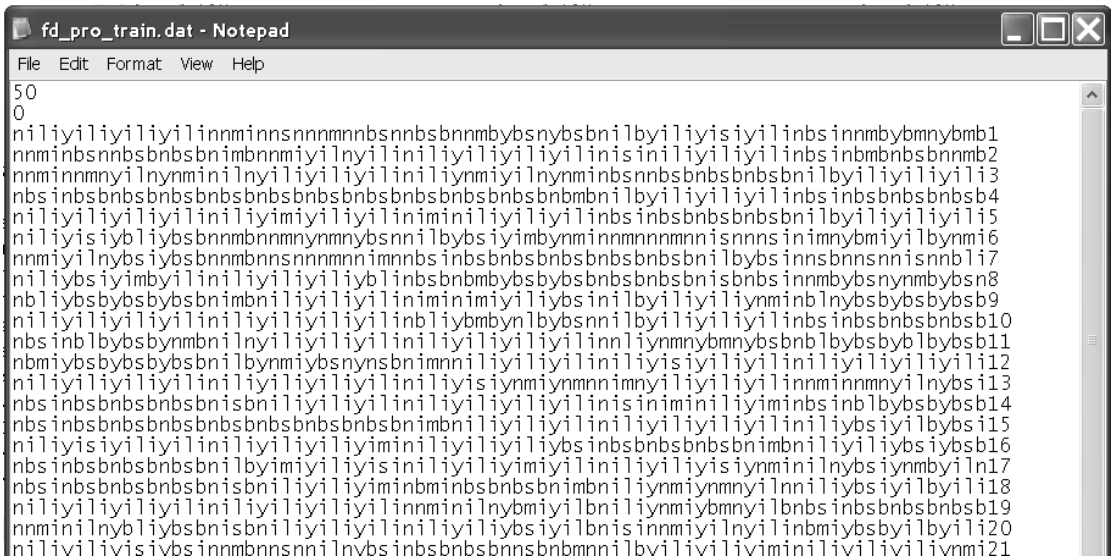



Figure 6.3: the part of the training data created using the probabilistic approach

Comparing this file to Figure 6.3, we can find out that the states of KEYWORD are the same, but the states of DURATION and IG are not the same.

### *6.3.2 Training the Network Using the Above Data Set*

In this step, we used the same prior probability values and procedure to train the network. The difference was that the training data was created using a different method. We do not repeat the procedure here. The CPDs are as follows:

| **Node IG** | **P(IG)** | **Number** |
|---|---|---|
| YES | 0.53 | 528 |
| NO | 0.47 | 472 |
| **Sum** | | 1000 |
| | | |
| **Node PKW** | **P(PKW)** | **Number** |
| INTEREST | 0.54 | 543 |
| NEUTRAL | 0.16 | 157 |
| BORING | 0.30 | 300 |
| **Sum** | | 1000 |
| | | |
| **Node D** | **P(D / KW)** | **Number** |
| LONG/INTEREST | 0.85 | 429 |
| MIDDLE/INTEREST | 0.08 | 40 |
| SHORT/INTEREST | 0.07 | 37 |
| **Sum** | | 506 |
| LONG/NEUTRAL | 0.07 | 11 |
| MIDDLE/NEUTRAL | 0.78 | 130 |
| SHORT/NEUTRAL | 0.15 | 25 |
| **Sum** | | 166 |
| LONG/BORING | 0.06 | 20 |
| MIDDLE/BORING | 0.13 | 41 |
| SHORT/BORING | 0.81 | 267 |
| **Sum** | | 328 |
| | | |
| **Node KW** | **P(KW / IG, PKW)** | **Number** |
| INTEREST/YES, INTEREST | 0.74 | 269 |
| NEUTRAL/YES, INTEREST | 0.15 | 55 |
| BORING/YES, INTEREST | 0.11 | 39 |
| **Sum** | | 363 |
| INTEREST/YES, NEUTRAL | 0.50 | 39 |
| NEUTRAL/YES, NEUTRAL | 0.23 | 18 |
| BORING/YES, NEUTRAL | 0.27 | 21 |
| **Sum** | | 78 |
| INTEREST/YES, BORING | 0.18 | 16 |
| NEUTRAL/YES, BORING | 0.18 | 16 |
| BORING/YES, BORING | 0.63 | 55 |
| **Sum** | | 87 |
| INTEREST/NO, INTEREST | 0.53 | 95 |
| NEUTRAL/NO, INTEREST | 0.16 | 28 |
| BORING/NO, INTEREST | 0.32 | 57 |
| **Sum** | | 180 |
| INTEREST/NO, NEUTRAL | 0.54 | 43 |
| NEUTRAL/NO, NEUTRAL | 0.25 | 20 |
| BORING/NO, NEUTRAL | 0.20 | 16 |
| **Sum** | | 79 |
| INTEREST/NO, BORING | 0.21 | 44 |
| NEUTRAL/NO, BORING | 0.14 | 29 |
| BORING/NO, BORING | 0.66 | 140 |
| **Sum** | | 213 |

Table 6.13: the CPDs of the node KW

Comparing Table 6.13 to Table 6.7, we can see the conditional probability distribution of node PKW has not been varied, and the sums of events are also the same because the CPD of the node PKW and the sum of every event has been fixed in the personal models we designed in the first step. However, the CPDs of other nodes were quite different due to the probabilistic approach used.

### *6.3.3 Testing the Network using SAME Data Set*

Similarly, we also needed to test the network using the training data set. The follow tables are the PDTs of the corresponding junction tree's nodes.

| KW | PDT |
|---|---|
| Interesting | 1.00000000 |
| Neutral | 1.00000000 |
| Boring | 1.00000000 |

Table 6.14: the PDT for node KW

| D | KW | PDT |
|---|---|---|
| Long | Interesting | 0.84767437 |
| Middle | Interesting | 0.07909899 |
| Short | Interesting | 0.07322664 |
| Long | Neutral | 0.06636966 |
| Middle | Neutral | 0.78285974 |
| Short | Neutral | 0.15077062 |
| Long | Boring | 0.06107894 |
| Middle | Boring | 0.12505943 |
| Short | Boring | 0.81386161 |

Table 6.15: the PDT for node D-KW

| KW | IG | PKW | PDT |
|---|---|---|---|
| Interesting | Yes | Interesting | 0.21243942 |
| Neutral | Yes | Interesting | 0.04343531 |
| Boring | Yes | Interesting | 0.03079908 |
| Interesting | Yes | Neutral | 0.04143302 |
| Neutral | Yes | Neutral | 0.01914579 |
| Boring | Yes | Neutral | 0.02230846 |
| Interesting | Yes | Boring | 0.02913089 |
| Neutral | Yes | Boring | 0.02915817 |
| Boring | Yes | Boring | 0.10009426 |
| Interesting | No | Interesting | 0.13526489 |
| Neutral | No | Interesting | 0.03987084 |
| Boring | No | Interesting | 0.08113617 |
| Interesting | No | Neutral | 0.04029904 |
| Neutral | No | Neutral | 0.01879123 |
| Boring | No | Neutral | 0.01500676 |
| Interesting | No | Boring | 0.02924752 |
| Neutral | No | Boring | 0.01928130 |
| Boring | No | Boring | 0.09305786 |

Table 6.16: the PDT for node KW-IG-PKW

Table 6.17 shows the resultant ratio of the test.

| | |
|---|---|
| The number of correct inferred state | 826 |
| The number of incorrect inferred state | 174 |
| The correct result ratio | 0.826 |

Table 6.17: the correct result ration of the probabilistic way

### *6.3.4 Retesting the Network Using another Different Set of Artificial Data*

We also created another different data set using the same way for testing the network again. Table 6.18 shows the result.

| | |
|---|---|
| The number of correct inferred state | 864 |
| The number of incorrect inferred state | 136 |
| The correct result ratio | 0.864 |

Table 6.18: the correct result ration of the probabilistic way

From those two results, we can see that with probabilistic data the network can still perform very accurately. To further prove this we conducted an additional evaluation step.

### *6.3.5 Retesting the Network Using the Different Data Combination*

In this step, we used the different data combination to test the network. We tested the models learned in evaluation one using the data created for evaluation two. The Bayesian network was trained on non-probabilistic data, but tested using probabilistic data, for a variety of parameters. And then we changed the roles, so that the network was trained on probabilistic data, but tested using non-probabilistic. Meanwhile, we also used two different prior probability tables. In the above evaluations, the prior probability was assigned the values designed in advance. This time, we created an equal prior probability table for each node. See Table 6.19.

| IG | P(IG) |
|---|---|
| YES | 0.50 |
| NO | 0.50 |

| KEYWORD | DURATION | P(D / KW) |
|---|---|---|
| INTERESTING | LONG | 0.33 |
| INTERESTING | MIDDLE | 0.33 |
| INTERESTING | SHORT | 0.34 |
| NEUTRAL | LONG | 0.33 |
| NEUTRAL | MIDDLE | 0.33 |
| NEUTRAL | SHORT | 0.34 |
| BORING | LONG | 0.33 |
| BORING | MIDDLE | 0.33 |
| BORING | SHORT | 0.34 |

| PRIOR KW | IG | KEYWORD | P(KW / IG, PKW) |
|---|---|---|---|
| INTERESTING | YES | INTERESTING | 0.33 |
| INTERESTING | YES | NEUTRAL | 0.33 |
| INTERESTING | YES | BORING | 0.34 |
| INTERESTING | NO | INTERESTING | 0.33 |
| INTERESTING | NO | NEUTRAL | 0.33 |
| INTERESTING | NO | BORING | 0.34 |
| NEUTRAL | YES | INTERESTING | 0.33 |
| NEUTRAL | YES | NEUTRAL | 0.33 |
| NEUTRAL | YES | BORING | 0.34 |
| NEUTRAL | NO | INTERESTING | 0.33 |
| NEUTRAL | NO | NEUTRAL | 0.33 |
| NEUTRAL | NO | BORING | 0.34 |
| BORING | YES | INTERESTING | 0.33 |
| BORING | YES | NEUTRAL | 0.33 |
| BORING | YES | BORING | 0.34 |
| BORING | NO | INTERESTING | 0.33 |
| BORING | NO | NEUTRAL | 0.33 |
| BORING | NO | BORING | 0.34 |

Table 6.19: the new prior pro tables for three nodes

Therefore, we could combine four kinds of different data to test the network again. See Table 6.20. “Prior Pro” means the prior probability which had two kinds, Old and New. The column “Training Data” shows the data that was used to train the network, while the column “Testing Data” shows that data that was used to test the network. All data had two types, non-probabilistic and probabilistic. Non-probabilistic data was generated by the rules, and the probabilistic one by the probabilistic method.

| Prior Pro | Training Data | Testing Data |
|---|---|---|
| Old | Non-probabilistic | Probabilistic |
| Old | probabilistic | Non-probabilistic |
| New | Non-probabilistic | Probabilistic |
| New | probabilistic | Non-probabilistic |

Table 6.20: the data combination for testing the network

The testing results are shown in Table 6.21.

| Prior Pro | Training Data | Testing Data | Resultant Ratio |
|---|---|---|---|
| Old | Non-probabilistic | probabilistic | 0.876 |
| Old | probabilistic | Non-probabilistic | 0.935 |
| New | Non-probabilistic | probabilistic | 0.872 |
| New | probabilistic | Non-probabilistic | 0.934 |

Table 6.21: the result of the data combination testing

### *6.3.6 Summary*

In this section we have performed a second evaluation to test if the network is general. To prove this point, we had to create another method which also can generate the artificial data according to the personal model designed in advance. We called the new method the probabilistic method. The data created by this way was the probabilistic data.

Firstly, we repeated the first five steps of Evaluation One. Similarly, we also obtained two accurate ratios which were 0.826 and 0.864. These ratios were slightly lower than those found in the first evaluation.

Secondly, we exchanged the roles of data, and created the new prior probability table to more fully test the network. Therefore, we made a data combination table, (Table 6.20), and used the different combination as the training and testing data. From Table 6.21, we found out that the change of prior probability hardly affected the accuracy of the result. Meanwhile, whether the prior probability was new or old, the results from the non-probabilistic training data were lower than those from the probabilistic training data.

In short, we could say the network is able to adapt to different conditions so that we suggest the network could be generalizable.

## 6.4 Evaluation Three: Is It Suitable for Other Circumstances?

In the third evaluation, we wanted to test the following points:

1. How robust is the network is at predicting under the simulative circumstance?
2. How robust is the matchmaker is at calculating the similarity of the simulative agents?
3. How does the network affect the agent's behavior?
4. How does the agent's behavior affect the ability of the network?

### *6.4.1 Presenting a Simulative Conference using the Simulator*

We designed a simulated conference to explore these four points. In the conference, the agents were allowed to watch the posters on the wall. The posters were attached with an infrared sensor which could communicate with the badges the agents were wearing. The badges communicated with the sensors and collected the data from it. The data included the time the agent spent on the poster, and the poster ID. The data would be transferred to the network when the agent left the current poster. The system figured out how much the agent was interested in this poster according to the data from the simulator. Therefore, the system had the capability to deliver the right information to the right agent. In short, the agent could obtain more information about topics of interest from the guide.

Meanwhile, when two agents encountered each other, the sensors on the badges could communicate with each other and exchange the badge ID. Then the badges sent the data to the server. The server could calculate the similarity between those two agents. According to the similarity, we could tell the relationship of them and give feedback. This could enhance the communication of agents.

We attached 20 infrared sensors to 20 posters. The range of the simulated sensors was 2 meters and 30 degrees. Meanwhile, each agent wore one badge which had an infrared sensor with range 1.5 meters and 30 degrees.

The main task of the simulator was to calculate the time spent watching and talking. When the agent was moving into the range of the sensors, the simulator started to record the time until the agent left from a conversation or watching a poster.

To simplify the task, we applied the rules and the probabilistic method to generate the time duration for each agent. The difference from the data generated by the simulator with the artificial data used in Evaluation One and Evaluation Two, was that the behavior of the agents was random and uncertain, because the computation of the agent's behavior was based on the current environment the agent is in. However, we cannot control the environment. For example, when the agents encounter each other, we do not know if they will start a conversation or perform a steering action. We could not know what the agents would do in advance.

### *6.4.2 Figuring Out the Interest Table for Every Agent*

After designing the virtual conference, we started the evaluation and as time passed, the interest tables were inferred in real time. When all interest tables were finished, we stopped the application and stored the result into the .dat file which contained the new interest tables.

### *6.4.3 Comparing the New Interest Tables to the Original Tables*

We designed the personal models which contained the original interest tables. We compared the new interest tables to the original one. The same procedure that we performed in last evaluation was used to work out an accurate ratio. See Table 6.22. The first row shows the number of the correct inferred interests in 1000 interests, in which the duration was determined by the rules; the second row presents the number of the correct inferred interests in which the duration was determined probabilistically.

| The Methods | The Accurate Ratio |
|---|---|
| Rules | 0.820 |
| Probabilistic data | 0.843 |

Table 6.22: the accurate ratio of the network performed

### *6.4.4 Calculating the Original Similarity of 50 Agents*

We used the matchmaker to calculate the original similarity according to the original interest tables. After that, we distinguished the relationship among the 50 agents based on the original similarity. The relationship state had three values which are FRIENDLY, the corresponding similarity is 15 - 20, NEUTRAL, the corresponding similarity is 9 – 14 and NOT FRIENDLY, the corresponding similarity is 2 – 8. See Table 6.23.

| Similarity | Relationship States |
|---|---|
| 2 – 8 | FRIENDLY |
| 9 – 14 | NEUTRAL |
| 15 - 20 | NOT FRIENDLY |

Table 6.23: the similarity and the relationship states

Among the 50 agents, there were 107 pairs of agents that could be friendly, and 301 pairs of neutral normal relationships. However, there were 204 pairs of agents that did not like each other.

### *6.4.5 Calculating the Similarity of Agents in Real Time*

Since the matchmaker calculated the similarity in real time and the interest tables were incomplete at beginning, the result could be not accurate at that time. However, as the network completed the interest tables of the agents, the matchmaker could work out the similarity more accurately. Therefore, we allowed for the similarity of the agents to be updated while the simulation was running.

In the same way, we calculated the relationship among the 50 agents based on the new similarity. By using the rules, there were 112 pairs of agents who could make friends with each other. In contrast, 195 pairs did not like each other. Finally, we had

314 pairs of agents who were in the middle level. On the other hand, by using the probabilistic approach there were not too many changes compared to the first one. 110 pairs were friends, 197 pairs were not friendly and 310 pairs were in the middle level. The result is shown in Table 6.24 and Table 6.25.

| Relationship States | Number |
|---|---|
| Friendly | 112 |
| Normal | 324 |
| Not Friendly | 195 |

Table 6.24: the new relationship distribution using the rules

| Relationship States | Number |
|---|---|
| Friendly | 110 |
| Normal | 310 |
| Not Friendly | 197 |

Table 6.25: the new relationship distribution using the probabilistic data

### *6.4.6 Summary*

This evaluation explored two goals. To make the evaluation more general, we applied two methods to calculate the duration. From the results of the network and the matchmaker, the effect was small. For example, the number of correctly inferred interests was 843 per 1000 when we used the probabilistic method to create the duration time. On the other hand, the number of the correct interests was 820 per 1000 when we used the rule based approach. Furthermore, from Table 6.24 and Table 6.25, we could make a conclusion that the matchmaker can work well whether we used the rules or the probabilistic method.

### How Robust is the Network at Predicting under the Simulated Circumstances?

The first goal was to prove the network could be robust at predicting under a more complex circumstance. In the first step, we designed and created a simulated conference for this goal. The followed two steps calculated the results which were 82%

using the rules and 84.3% using the probabilistic method respectively. These two ratios could prove the network has the capability to work well under a more complex environment.

### How Robust is the Matchmaker at Calculating the Similarity of the Simulated Agents?

The second goal was to prove the matchmaker could work well. First, we calculated the original similarity scores based on the personal models. Secondly, we calculated the new similarity scores according to the new personal models which were inferred by the network in real time. Finally, we could obtain a new and an original relationship state among the 50 agents through groupings based on the new and original similarity.

The original relationship distribution is shown in Figure 2.8. Table 6.24 and Table 6.25 show the new relationship distributions. We could see that the number of the NORMAL and FRIENDLY relationships increased a little bit, from 301 to 324 and 310, 107 to 112 and 110 respectively. The number of NOT FRIENDLY decreased by 9 and 7. In short, although there were some increases and decreases, the matchmaker still could calculate most similarity scores correctly.

| Relationship | Number (pair) |
|---|---|
| FRIENDLY | 107 |
| NORMAL | 301 |
| NOT FRIENDLY | 204 |

Table 6.26: the original relationship distribution

### How Did the Network Affect the Agent's Behaviour?

At the beginning of simulation, the interest tables were incomplete. Thus, we could see the duration of talk between agents was not long, mostly 1-2 minutes of simulated time. As the interest table was being completed, we could find some pairs of agents staying together for over 6-7 minutes of simulated time. This shows that when two agents obtained the feedback from the central server that they had common interests, they would talk longer about those interests. Therefore, in the simulation the ability of

the network directly affects the communication of agents. It can help agents get into a deep conversation about their common interests. However, this would still need to be verified by testing with smart badge hardware in a real environment.

We still suppose the network could affect the agent's behaviour from two aspects in the real environment. One is to enhance the communication between agents. Another one is to provide a direction to the agents according the inferred interest.

Since the badge is able to provide the information about another person before they start a conversation, the conversation between participants can be enhanced. For example, if the badge displays their common interests, then the time spent talking could be longer than in the situation without the badge. Moreover, the interests displays are the results of the matchmaker and the network. Therefore, we suppose that the matchmaker and the network could affect the agent behaviour.

Not only that, the result of the Bayesian network can also provide a direction to agents. This is because the central server has already figured out the interests of every agent and it also knows where other relevant objects are, and when other relevant objects will be shown. This kind of information can be sent to agents as feedback. After agents get the feedback like this, they should make a plan of next steps according to the feedback. Therefore, we can say the network can help agents obtain more information which are relevant to their interests. Although this point has not been implemented in the current version of the simulator, it could be in the future.

### How Did the Agent's Behaviour Affect the Ability of the Network?

We can say the virtual agent's behaviour can extremely affect the ability of the network. Firstly, in the graph model of the network, there are two nodes, Duration and Prior Keyword. Duration means how long the agent spends on an object. The Prior Keyword means how much interest the agent has in the last object he looked at. Those both are agent's properties which determine the ability of the network. Furthermore, if the badge is able to record the talk time, then the talk duration could be a new node of network to make the network more accurate.

In short, we can use the simulator to prove the ability of Bayesian network to affect the agent's behaviour. Meanwhile, we also know the agent's behaviour can extremely affect the ability of the network. However, this was tested in an ideal environment. The simulator created an ideal environment for evaluation, in which we assumed the virtual agents are rational, for example, and can talk to one another for a long time if it thinks that one is a friend. However, in the real world, the environment should be changed. There exist other conditions to constrain agents, for example, we have to consider the personal characters of agents. Even if a talkative person encounters a total stranger, he probably keeps talking for a long duration. We will analyse what will happen if the badge is used in a real world example in the next section.

## 6.5 Summary

In this chapter, we firstly designed a personal model for 50 virtual agents. For producing the training data set, we created artificial data using two different methods; rules-based and probabilistic. All artificial training data sets are based on the 50 agent's personal models. Section 6.2 used two different ways to evaluate the network in order to prove the network is able to work well. Section 6.3 presented the evaluation which is to prove the network is general. Two groups of testing and training data were exchanged to test the network, and the network still performed effectively. Section 6.4 described the third evaluation which created a virtual conference circumstance using the simulator and then verified the network and the matchmaker at the same time.

We could find some usable points from the evaluation. Firstly, artificial data is good to use when collecting real training data is impossible. However just using artificial data is not enough to prove the tested object is generalizable. Thus, we need to design at least two types of approaches for this point. Secondly, care needs to be taken in applying artificial data to train and test the object. For example, in our evaluation, we firstly trained the network using rule-based artificial data, and then tested it using probabilistic artificial data. For more accuracy, we changed the roles of two kind of artificial data in the second step. Finally, the use of a simulation is also a good idea. Although we can also use other approaches to implement the evaluation, we still chose

the simulation. The simulation not only provides a simulative environment which is similar to the real world, but we also revise the parameters of the simulation in order to verify how the network works under different conditions.

There were two problems which have not been solved yet since there was not enough time. Those problems could invalidate the result of the evaluation. Here, we present them, and we will discuss them in next chapter in detail.

To test and train the network, the data set we used was generated by two approaches. These approaches were different, but the agent's behaviour they represented was similar. If the training data was from a population that behaved differently to the test population, does the network have a good performance?

Another problem which can invalidate the result is that the level of randomness turned out to be much larger than I modeled. The two approaches used just one level of randomness to generate the artificial data. If we use other level of randomness, does the network have a good performance?

### Discussion

The tests and evaluation were implemented in an ideal simulative environment. We need an in-depth analysis whether the network and matchmaker still can be efficiently in real world.

For testing the network, the data set we used was generated by two approaches. These approaches were different, but the agent's behaviour they represented was similar. In those data sets, the agents looked at the posters one by one. The training data was from a population that behaved differently to the test population, which may affect the result, but we cannot know how much the effect is because it is based on how different the behaviour is. If the agents show normal behaviour, the effect should not be significant because we made an assumption that we do not care about the visiting order. The data sets were generated based on the assumptions.

Another situation which could affect the results is that the level of randomness turned out to be much larger than I modeled. We have not tested how large the randomness is when the network performs with less accuracy. However, the randomness should be reasonable.

At first, let's us see what the common and different points of the real word and the simulation are. The common points are that:

- The agents in both have human basic behaviour, such as steering, obstacles avoidance, and following a path. Moreover, they also can look at the posters on the wall, and stop for a talk with other people.
- In both environments, the objects are grouped according to the contents they represent. In the simulated environment, we have 5 rooms which contain 5 various kinds of objects.

The different points are that:

- In the real world, there should be many available paths the agent can follow, but there is only one path in the simulated world.
- In real world, the number of participants of a talk can be over two, but we limited the maximum number to two in the simulated world.
- We do not consider the personal characters of agents in the simulated environment, but these in fact exist in the real world.

There should be other common points and different points. Here, we just covered the above main points.

Actually, people in a real conference or museum still follow a path to visit the objects one by one. The path is not fixed and it can be arbitrary, however there is still a rule which is that people normally visit the object based on an order, but not jump to do that. For example, when a person enters a hall, he would visit the objects in this hall one by one. It is impossible that he finishes looking at the first object in the first hall, and then goes to another hall to look at the object in the second hall. After that, he comes back to the first hall for looking at the second object. Therefore, we can say the design of the path cannot affect the result too much as long as the path is designed rationally.

The number of participants in a conversation can be more than two in the real world. If there are three or four persons gathered together, then it leads to their badges exchanging the data more than once. However, it does not affect what you can read from your badge. You still know who is Person A with common interest 1, and interest 2, who is Person B with common interest 2 and common interest 3. However, if in the future we need to record the talk time it will become a problem. Since the infrared sensor can be affected by the noise, the badge can produce a wrong record. For example, when you are talking with Person A, but your badge is facing Person B's badge due to some reasons, your badge will record the talk time as the duration of a talk of you and Person B, but not Person A.

The personal characters of participants will cause more negative effect for the result compared to the above points. For example, if Person A likes to skip a poster, but Person B likes to peruse a poster, then although they both are interested in an object, they spend an absolutely different duration time looking at the object. Thus it causes an error in the result of the Bayesian network. Sometime, this kind of problem can make the network invalid in the real world.

Furthermore, there are other assumptions which might invalidate the network in real world. For example, if a conference does not provide the objects to represent, then the network cannot use to infer the interest of the participants in this kind of environment. Finally, if the distance between the participants and the objects is over the specific distance range of the infrared sensor, then it also causes the network to be invalid.

# Chapter 7

# CONCLUSION AND THE FUTURE WORK

In this research we have developed software for a Smart Badge system. The system contains three parts; one is the hardware we call Smart Badge which can collect data from infrared sensors attached to objects, and deliver useful data to a central server machine. The second part is the central server machine which communicates with the badges and analyses data from the badges using a Bayesian network. The third one is the security protocol for data communication. To achieve this purpose, we have proposed five objectives in Chapter 1 as follows:

1. Smart Badge hardware design;
2. Developing a simulation application;
3. Developing an intelligent central server;
4. Building a security protocol for data communication;
5. Evaluating the effectiveness of the intelligent server using the simulation.

Objective 1 and Objective 4 have been performed by other two masters students. In this thesis, we have proposed an evaluation system for Smart Badge. The evaluation system has been used to test and evaluate the Smart Badge's central server.

To achieve Object 2, we developed a simulator that created a virtual conference environment with 50 virtual agents with the smart badges. We allowed the virtual agents to make conversation and look at posters in a simulated environment. The simulator recorded the time spent on these behaviors. Each agent could send the collected data to the central server according to the agent's current state. The agents could determine what the next action is based on the information from the central server. Chapter 3 has described the design and implementation. This enabled us to repeatedly test our algorithms and it avoided the expense of having to develop real working badges. Through this simulator, we can apply groups of different testing data, so that we are able to perform a complete evaluation of the system.

To achieve Object 3, we used a Bayesian network in the main inference module and a revised profile matching algorithm of matchmaker in our central server. In Chapter 4, we designed the structure of Bayesian network. We discussed the design of a network, specific to our needs. The data we could obtain from the badges is time spent on watching and talking, and the position of the agents. We also knew the agent's historical behavior and the known interests. According to these data, we can figure out the state of every node in the network. The node we were interested in was the current keyword. Finally, we could calculate the interest level of the current keyword through updating the CPT of every node. Furthermore, to figure out the similarity of agents, we revised a profile matching algorithm. The original algorithm fits the web-based date system. We revised this in order to satisfy our requirement. The biggest advantage is that the algorithm can accept the incomplete profile of the agent. The computation of the similarity may enhance the communication of the agents when it knows they have similar interests. For example, the server may announce to an agent in advance about another agent's interests or common ideas before they begin to talk. We revised the matchmaking algorithm in order to fit our needs. More detailed information has been described in Chapter 4.

To achieve Object 5, we made the evaluation plan presented in Chapter 5. We used a method called artificial data to generate the testing and training data. Chapter 5 also explained the reason why we have to use it, and how we use it.

Chapter 6 covered the procedure and the result of the evaluation. We have performed three evaluations. The first evaluation was to verify whether the network can work. We used two different artificial data to train and test the network. The result showed the network is able to perform a great job. The two correct ratios were over 80%. The second evaluation was to verify whether the network can be "general". Here, "general" means the trained network by a kind of data should fit other kinds of data. Thus, we exchanged the role of these two kinds of data; trained the network using the first one, and tested it using the second one. The correct ratio was also over 80%. In the third evaluation, we used the simulator to create the testing data. Since the agents' actions are random, the simulator may generate the testing data randomly. We also used two

sorts of methods to generate the testing data. Meanwhile, we tested the matchmaker using the simulator.

We discussed the interaction between the network and the agent's behavior in the simulative environment and in the real life. Although we could not perform an experiment in the real environment, the performance in the simulated environment is very encouraging. However, the simulative environment is not same as the real one, so we illustrated the common and different points between those two environments. We also discussed what would happen if the Smart Badge was used in the real environment, and what situations could invalid the whole system

In this thesis, we do not design a new inference algorithm and a new matchmaking algorithm but apply a novel combination of Bayesian network and a profile matching algorithm so that we could build a server for a Smart Badge system. We created a novel way to use an evaluation system to test and evaluate Smart Badge technology. We performed an experiment in the simulated environment. The performance of the simulated Smart Badge satisfied the requirements we mentioned in Chapter 5. Through comparing the real environment to the simulated one, it seems that the Smart Badge could perform well with our Bayesian approach. However, due to some shortcomings we discussed in summary of Chapter 6, the efficiency of Smart Badge could be negatively affected.

## 7.1 The Future Work

Firstly, we need to solve two problems that can invalid the result we mentioned in Chapter 6. The first problem is that the training data was from a population that behaved differently to the test population. The situation does affect the result. To solve this problem, we would update the simulation application in order to create different agent behaviours. Therefore, we could collect the training data from those agents who behave differently, and train the network using this training data.

Another problem is that the different level of randomness affects the result. To solve this problem, we would design a group of randomness which are at different levels for the two approaches we used to generate the artificial data. After that, we would repeat Evaluation One according to the different level of randomness. This would show how well the Bayesian network we use is able to capture the information required to predict their likes, and how resistant to “noise” it is.

Secondly, the most immediate future research is to verify (or refute) the assumptions. We also need to evaluate what happens when the model is wrong. We would run the simulation for different parameters (e.g. level of randomness). We would create different combinations in order to make the “wrong” parameters. For example, when Agent 1, who likes A, but it is not on his badge encounters Agent 2, who also likes A, which is not on his badge either, what will happen? Or Agent 1 does not like A, but it is on badge (e.g. the network has incorrectly induced that he likes A). We create these combinations, and run the simulation with them. If either the network or the matchmaker is inaccurate, does the system still gradually get the right people together, or does it simply collapse altogether and become ineffective?

Thirdly, we need to evaluate the effect between the inference mechanism and the matchmaker mechanism when one mechanism is wrong. For example, if the inference mechanism figures out the wrong result, how much will this affect the matchmaker?. Or, if the matchmaker is wrong, how will this affect the inference mechanism? Finally, we look forward to updating the structure of Bayesian network. We use conversation time duration as a node of the network. I suppose this way would be more accurate to infer the interest of people.

Fourthly, if we could achieve the above three points. The fourth future work is to update the Bayesian network we used. We could add two nodes to the current network. One is the time spent on the conversation which can be collected be the infrared sensors. Another one is the similarity of the participants of the conversation. We used to try to construct a network including those two nodes. However, we found that there was an error about the causal links among the nodes. I still think the two

new evidences could enhance the inference accuracy if we could find out a correct structure for the network.

Finally, several other issues were noted that could be explored for future work:

1. Voice Recognition. The hardware might recognize the voice of the attendees, and the software could know what the topic they are talking about, in order to infer the interests or ideas more correctly.
2. Intelligent Feedback. Although the central server can infer the attendee interests, it still does not have the capability to deliver the relative information to the attendees. Therefore, a new module could be designed and implemented in the server to analyze what the relative information is according to the inferred interests.
3. The Interaction of the Network and the Matchmaker. So far, the input of the matchmaker algorithm is the output of the network, so we could say the network can affect the matchmaker. We could try to find how the similarity that is the result of the matchmaker probably supports the inference of the network.

APPENDIX A

# The Artificial Agents' Registration Information

| ID | COSC01 | | | ID | COSC02 | |
|---|---|---|---|---|---|---|
| **The KW Table** | **ID** | **Level** | | **The KW Table** | **ID** | **Level** |
| | 1 | 0.8 | | | 1 | 0.4 |
| | 2 | 0.7 | | | 2 | 0.2 |
| | 3 | 0.9 | | | 3 | 0.3 |
| | 4 | 0.6 | | | 4 | 0.2 |
| | 5 | 0.5 | | | 5 | 0.7 |
| | 6 | 0.5 | | | 6 | 0.5 |
| | 7 | 0.4 | | | 7 | 0.6 |
| | 8 | 0.3 | | | 8 | 0.7 |
| | 9 | 0.3 | | | 9 | 0.9 |
| | 10 | 0.4 | | | 10 | 0.9 |
| | 11 | 0.2 | | | 11 | 0.8 |
| | 12 | 0.3 | | | 12 | 0.6 |
| | 13 | 0.8 | | | 13 | 0.8 |
| | 14 | 0.9 | | | 14 | 0.7 |
| | 15 | 0.7 | | | 15 | 0.8 |
| | 16 | 0.8 | | | 16 | 0.7 |
| | 17 | 0.3 | | | 17 | 0.3 |
| | 18 | 0.4 | | | 18 | 0.3 |
| | 19 | 0.3 | | | 19 | 0.2 |
| | 20 | 0.2 | | | 20 | 0.4 |
| **ID** | COSC03 | | | **ID** | COSC04 | |
| **The KW Table** | **ID** | **Level** | | **The KW Table** | **ID** | **Level** |
| | 1 | 0.5 | | | 1 | 0.2 |
| | 2 | 0.4 | | | 2 | 0.1 |
| | 3 | 0.6 | | | 3 | 0.2 |
| | 4 | 0.4 | | | 4 | 0.3 |
| | 5 | 0.9 | | | 5 | 0.1 |
| | 6 | 0.7 | | | 6 | 0.8 |
| | 7 | 0.9 | | | 7 | 0.2 |
| | 8 | 0.8 | | | 8 | 0.9 |
| | 9 | 0.6 | | | 9 | 0.2 |
| | 10 | 0.5 | | | 10 | 0.3 |
| | 11 | 0.7 | | | 11 | 0.2 |
| | 12 | 0.5 | | | 12 | 0.7 |
| | 13 | 0.2 | | | 13 | 0.4 |
| | 14 | 0.2 | | | 14 | 0.7 |
| | 15 | 0.1 | | | 15 | 0.3 |
| | 16 | 0.3 | | | 16 | 0.8 |
| | 17 | 0.8 | | | 17 | 0.8 |
| | 18 | 0.6 | | | 18 | 0.6 |
| | 19 | 0.9 | | | 19 | 0.7 |
| | 20 | 0.7 | | | 20 | 0.4 |

| ID | COSC05 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.9 |
| | 2 | 0.8 |
| | 3 | 0.1 |
| | 4 | 0.2 |
| | 5 | 0.8 |
| | 6 | 0.7 |
| | 7 | 0.9 |
| | 8 | 0.7 |
| | 9 | 0.6 |
| | 10 | 0.5 |
| | 11 | 0.4 |
| | 12 | 0.3 |
| | 13 | 0.4 |
| | 14 | 0.1 |
| | 15 | 0.2 |
| | 16 | 0.1 |
| | 17 | 0.7 |
| | 18 | 0.6 |
| | 19 | 0.8 |
| | 20 | 0.6 |

| ID | COSC06 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.6 |
| | 2 | 0.8 |
| | 3 | 0.2 |
| | 4 | 0.1 |
| | 5 | 0.5 |
| | 6 | 0.5 |
| | 7 | 0.4 |
| | 8 | 0.3 |
| | 9 | 0.8 |
| | 10 | 0.1 |
| | 11 | 0.9 |
| | 12 | 0.5 |
| | 13 | 0.5 |
| | 14 | 0.4 |
| | 15 | 0.6 |
| | 16 | 0.5 |
| | 17 | 0.9 |
| | 18 | 0.2 |
| | 19 | 0.9 |
| | 20 | 0.5 |

| ID | COSC07 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.5 |
| | 2 | 0.9 |
| | 3 | 0.2 |
| | 4 | 0.1 |
| | 5 | 0.5 |
| | 6 | 0.5 |
| | 7 | 0.4 |
| | 8 | 0.7 |
| | 9 | 0.2 |
| | 10 | 0.1 |
| | 11 | 0.2 |
| | 12 | 0.3 |
| | 13 | 0.8 |
| | 14 | 0.1 |
| | 15 | 0.5 |
| | 16 | 0.2 |
| | 17 | 0.5 |
| | 18 | 0.4 |
| | 19 | 0.7 |
| | 20 | 0.3 |

| ID | COSC08 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.9 |
| | 2 | 0.1 |
| | 3 | 0.9 |
| | 4 | 0.4 |
| | 5 | 0.6 |
| | 6 | 0.7 |
| | 7 | 0.5 |
| | 8 | 0.1 |
| | 9 | 0.8 |
| | 10 | 0.2 |
| | 11 | 0.3 |
| | 12 | 0.1 |
| | 13 | 0.4 |
| | 14 | 0.1 |
| | 15 | 0.9 |
| | 16 | 0.2 |
| | 17 | 0.4 |
| | 18 | 0.3 |
| | 19 | 0.8 |
| | 20 | 0.2 |

| ID | COSC09 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.3 |
| | 2 | 0.1 |
| | 3 | 0.2 |
| | 4 | 0.9 |
| | 5 | 0.4 |
| | 6 | 0.8 |
| | 7 | 0.7 |
| | 8 | 0.9 |
| | 9 | 0.6 |
| | 10 | 0.8 |
| | 11 | 0.9 |
| | 12 | 0.3 |
| | 13 | 0.8 |
| | 14 | 0.7 |
| | 15 | 0.8 |
| | 16 | 0.5 |
| | 17 | 0.2 |
| | 18 | 0.5 |
| | 19 | 0.2 |
| | 20 | 0.3 |

| ID | COSC10 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.8 |
| | 2 | 0.9 |
| | 3 | 0.4 |
| | 4 | 0.8 |
| | 5 | 0.8 |
| | 6 | 0.9 |
| | 7 | 0.1 |
| | 8 | 0.6 |
| | 9 | 0.3 |
| | 10 | 0.2 |
| | 11 | 0.5 |
| | 12 | 0.3 |
| | 13 | 0.9 |
| | 14 | 0.7 |
| | 15 | 0.9 |
| | 16 | 0.7 |
| | 17 | 0.3 |
| | 18 | 0.2 |
| | 19 | 0.1 |
| | 20 | 0.3 |

| ID | COSC11 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.2 |
| | 2 | 0.1 |
| | 3 | 0.2 |
| | 4 | 0.4 |
| | 5 | 0.8 |
| | 6 | 0.6 |
| | 7 | 0.7 |
| | 8 | 0.9 |
| | 9 | 0.9 |
| | 10 | 0.8 |
| | 11 | 0.9 |
| | 12 | 0.8 |
| | 13 | 0.4 |
| | 14 | 0.5 |
| | 15 | 0.3 |
| | 16 | 0.3 |
| | 17 | 0.2 |
| | 18 | 0.1 |
| | 19 | 0.3 |
| | 20 | 0.1 |

| ID | COSC12 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.1 |
| | 2 | 0.2 |
| | 3 | 0.9 |
| | 4 | 0.3 |
| | 5 | 0.6 |
| | 6 | 0.4 |
| | 7 | 0.3 |
| | 8 | 0.5 |
| | 9 | 0.7 |
| | 10 | 0.6 |
| | 11 | 0.8 |
| | 12 | 0.9 |
| | 13 | 0.9 |
| | 14 | 0.7 |
| | 15 | 0.8 |
| | 16 | 0.6 |
| | 17 | 0.1 |
| | 18 | 0.2 |
| | 19 | 0.4 |
| | 20 | 0.6 |

| ID | COSC13 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.8 |
| | 2 | 0.9 |
| | 3 | 0.8 |
| | 4 | 0.6 |
| | 5 | 0.5 |
| | 6 | 0.4 |
| | 7 | 0.2 |
| | 8 | 0.9 |
| | 9 | 0.6 |
| | 10 | 0.7 |
| | 11 | 0.5 |
| | 12 | 0.4 |
| | 13 | 0.9 |
| | 14 | 0.8 |
| | 15 | 0.7 |
| | 16 | 0.9 |
| | 17 | 0.5 |
| | 18 | 0.4 |
| | 19 | 0.6 |
| | 20 | 0.3 |

| ID | COSC14 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.1 |
| | 2 | 0.2 |
| | 3 | 0.3 |
| | 4 | 0.2 |
| | 5 | 0.8 |
| | 6 | 0.7 |
| | 7 | 0.8 |
| | 8 | 0.6 |
| | 9 | 0.9 |
| | 10 | 0.7 |
| | 11 | 0.8 |
| | 12 | 0.9 |
| | 13 | 0.9 |
| | 14 | 0.9 |
| | 15 | 0.7 |
| | 16 | 0.7 |
| | 17 | 0.3 |
| | 18 | 0.2 |
| | 19 | 0.2 |
| | 20 | 0.3 |

| ID | COSC15 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.2 |
| | 2 | 0.1 |
| | 3 | 0.1 |
| | 4 | 0.3 |
| | 5 | 0.2 |
| | 6 | 0.3 |
| | 7 | 0.1 |
| | 8 | 0.3 |
| | 9 | 0.9 |
| | 10 | 0.8 |
| | 11 | 0.7 |
| | 12 | 0.6 |
| | 13 | 0.9 |
| | 14 | 0.9 |
| | 15 | 0.6 |
| | 16 | 0.9 |
| | 17 | 0.2 |
| | 18 | 0.9 |
| | 19 | 0.1 |
| | 20 | 0.3 |

| ID | COSC16 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.8 |
| | 2 | 0.9 |
| | 3 | 0.9 |
| | 4 | 0.6 |
| | 5 | 0.8 |
| | 6 | 0.9 |
| | 7 | 0.7 |
| | 8 | 0.6 |
| | 9 | 0.9 |
| | 10 | 0.7 |
| | 11 | 0.8 |
| | 12 | 0.3 |
| | 13 | 0.2 |
| | 14 | 0.3 |
| | 15 | 0.2 |
| | 16 | 0.9 |
| | 17 | 0.6 |
| | 18 | 0.9 |
| | 19 | 0.1 |
| | 20 | 0.3 |

| ID | COSC17 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.1 |
| | 2 | 0.2 |
| | 3 | 0.1 |
| | 4 | 0.3 |
| | 5 | 0.9 |
| | 6 | 0.9 |
| | 7 | 0.8 |
| | 8 | 0.6 |
| | 9 | 0.8 |
| | 10 | 0.7 |
| | 11 | 0.6 |
| | 12 | 0.8 |
| | 13 | 0.7 |
| | 14 | 0.6 |
| | 15 | 0.7 |
| | 16 | 0.5 |
| | 17 | 0.2 |
| | 18 | 0.3 |
| | 19 | 0.4 |
| | 20 | 0.4 |

| ID | COSC18 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.1 |
| | 2 | 0.2 |
| | 3 | 0.1 |
| | 4 | 0.4 |
| | 5 | 0.8 |
| | 6 | 0.9 |
| | 7 | 0.4 |
| | 8 | 0.7 |
| | 9 | 0.8 |
| | 10 | 0.3 |
| | 11 | 0.2 |
| | 12 | 0.1 |
| | 13 | 0.9 |
| | 14 | 0.6 |
| | 15 | 0.5 |
| | 16 | 0.4 |
| | 17 | 0.6 |
| | 18 | 0.2 |
| | 19 | 0.7 |
| | 20 | 0.8 |

| ID | COSC19 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.8 |
| | 2 | 0.9 |
| | 3 | 0.9 |
| | 4 | 0.8 |
| | 5 | 0.9 |
| | 6 | 0.4 |
| | 7 | 0.7 |
| | 8 | 0.6 |
| | 9 | 0.5 |
| | 10 | 0.4 |
| | 11 | 0.9 |
| | 12 | 0.7 |
| | 13 | 0.8 |
| | 14 | 0.1 |
| | 15 | 0.2 |
| | 16 | 0.5 |
| | 17 | 0.3 |
| | 18 | 0.3 |
| | 19 | 0.8 |
| | 20 | 0.2 |

| ID | COSC20 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.5 |
| | 2 | 0.6 |
| | 3 | 0.3 |
| | 4 | 0.1 |
| | 5 | 0.8 |
| | 6 | 0.9 |
| | 7 | 0.8 |
| | 8 | 0.6 |
| | 9 | 0.9 |
| | 10 | 0.8 |
| | 11 | 0.1 |
| | 12 | 0.8 |
| | 13 | 0.9 |
| | 14 | 0.5 |
| | 15 | 0.7 |
| | 16 | 0.9 |
| | 17 | 0.2 |
| | 18 | 0.3 |
| | 19 | 0.4 |
| | 20 | 0.6 |

| ID | COSC21 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.8 |
| | 2 | 0.9 |
| | 3 | 0.9 |
| | 4 | 0.3 |
| | 5 | 0.5 |
| | 6 | 0.4 |
| | 7 | 0.9 |
| | 8 | 0.2 |
| | 9 | 0.3 |
| | 10 | 0.2 |
| | 11 | 0.5 |
| | 12 | 0.3 |
| | 13 | 0.9 |
| | 14 | 0.7 |
| | 15 | 0.6 |
| | 16 | 0.7 |
| | 17 | 0.8 |
| | 18 | 0.9 |
| | 19 | 0.7 |
| | 20 | 0.4 |

| ID | COSC22 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.5 |
| | 2 | 0.9 |
| | 3 | 0.4 |
| | 4 | 0.8 |
| | 5 | 0.8 |
| | 6 | 0.3 |
| | 7 | 0.9 |
| | 8 | 0.6 |
| | 9 | 0.6 |
| | 10 | 0.7 |
| | 11 | 0.5 |
| | 12 | 0.4 |
| | 13 | 0.9 |
| | 14 | 0.5 |
| | 15 | 0.9 |
| | 16 | 0.7 |
| | 17 | 0.8 |
| | 18 | 0.7 |
| | 19 | 0.4 |
| | 20 | 0.8 |

| ID | COSC23 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.2 |
| | 2 | 0.3 |
| | 3 | 0.4 |
| | 4 | 0.8 |
| | 5 | 0.7 |
| | 6 | 0.7 |
| | 7 | 0.9 |
| | 8 | 0.6 |
| | 9 | 0.2 |
| | 10 | 0.7 |
| | 11 | 0.7 |
| | 12 | 0.9 |
| | 13 | 0.7 |
| | 14 | 0.8 |
| | 15 | 0.5 |
| | 16 | 0.8 |
| | 17 | 0.5 |
| | 18 | 0.8 |
| | 19 | 0.4 |
| | 20 | 0.1 |

| ID | COSC24 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.2 |
| | 2 | 0.3 |
| | 3 | 0.7 |
| | 4 | 0.2 |
| | 5 | 0.6 |
| | 6 | 0.5 |
| | 7 | 0.7 |
| | 8 | 0.5 |
| | 9 | 0.5 |
| | 10 | 0.7 |
| | 11 | 0.4 |
| | 12 | 0.6 |
| | 13 | 0.3 |
| | 14 | 0.4 |
| | 15 | 0.2 |
| | 16 | 0.4 |
| | 17 | 0.9 |
| | 18 | 0.8 |
| | 19 | 0.7 |
| | 20 | 0.6 |

| ID | COSC25 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.2 |
| | 2 | 0.1 |
| | 3 | 0.3 |
| | 4 | 0.4 |
| | 5 | 0.7 |
| | 6 | 0.3 |
| | 7 | 0.3 |
| | 8 | 0.9 |
| | 9 | 0.7 |
| | 10 | 0.6 |
| | 11 | 0.7 |
| | 12 | 0.5 |
| | 13 | 0.7 |
| | 14 | 0.4 |
| | 15 | 0.6 |
| | 16 | 0.8 |
| | 17 | 0.5 |
| | 18 | 0.7 |
| | 19 | 0.8 |
| | 20 | 0.4 |

| ID | COSC26 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.8 |
| | 2 | 0.9 |
| | 3 | 0.7 |
| | 4 | 0.8 |
| | 5 | 0.2 |
| | 6 | 0.1 |
| | 7 | 0.3 |
| | 8 | 0.1 |
| | 9 | 0.4 |
| | 10 | 0.6 |
| | 11 | 0.5 |
| | 12 | 0.6 |
| | 13 | 0.9 |
| | 14 | 0.2 |
| | 15 | 0.7 |
| | 16 | 0.3 |
| | 17 | 0.2 |
| | 18 | 0.1 |
| | 19 | 0.3 |
| | 20 | 0.9 |

| ID | COSC27 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.1 |
| | 2 | 0.2 |
| | 3 | 0.5 |
| | 4 | 0.6 |
| | 5 | 0.9 |
| | 6 | 0.7 |
| | 7 | 0.6 |
| | 8 | 0.9 |
| | 9 | 0.4 |
| | 10 | 0.9 |
| | 11 | 0.8 |
| | 12 | 0.8 |
| | 13 | 0.9 |
| | 14 | 0.7 |
| | 15 | 0.9 |
| | 16 | 0.8 |
| | 17 | 0.2 |
| | 18 | 0.1 |
| | 19 | 0.1 |
| | 20 | 0.5 |

| ID | COSC28 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.8 |
| | 2 | 0.9 |
| | 3 | 0.7 |
| | 4 | 0.8 |
| | 5 | 0.6 |
| | 6 | 0.9 |
| | 7 | 0.5 |
| | 8 | 0.6 |
| | 9 | 0.5 |
| | 10 | 0.6 |
| | 11 | 0.3 |
| | 12 | 0.7 |
| | 13 | 0.9 |
| | 14 | 0.5 |
| | 15 | 0.9 |
| | 16 | 0.7 |
| | 17 | 0.8 |
| | 18 | 0.7 |
| | 19 | 0.9 |
| | 20 | 0.3 |

| ID | COSC29 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.8 |
| | 2 | 0.9 |
| | 3 | 0.7 |
| | 4 | 0.9 |
| | 5 | 0.8 |
| | 6 | 0.9 |
| | 7 | 0.7 |
| | 8 | 0.9 |
| | 9 | 0.3 |
| | 10 | 0.4 |
| | 11 | 0.2 |
| | 12 | 0.5 |
| | 13 | 0.4 |
| | 14 | 0.4 |
| | 15 | 0.6 |
| | 16 | 0.7 |
| | 17 | 0.2 |
| | 18 | 0.5 |
| | 19 | 0.9 |
| | 20 | 0.1 |

| ID | COSC30 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.2 |
| | 2 | 0.5 |
| | 3 | 0.4 |
| | 4 | 0.1 |
| | 5 | 0.9 |
| | 6 | 0.9 |
| | 7 | 0.8 |
| | 8 | 0.6 |
| | 9 | 0.5 |
| | 10 | 0.2 |
| | 11 | 0.7 |
| | 12 | 0.4 |
| | 13 | 0.6 |
| | 14 | 0.1 |
| | 15 | 0.9 |
| | 16 | 0.8 |
| | 17 | 0.4 |
| | 18 | 0.4 |
| | 19 | 0.1 |
| | 20 | 0.9 |

| ID | COSC31 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.4 |
| | 2 | 0.6 |
| | 3 | 0.1 |
| | 4 | 0.3 |
| | 5 | 0.8 |
| | 6 | 0.9 |
| | 7 | 0.7 |
| | 8 | 0.6 |
| | 9 | 0.9 |
| | 10 | 0.6 |
| | 11 | 0.7 |
| | 12 | 0.8 |
| | 13 | 0.2 |
| | 14 | 0.1 |
| | 15 | 0.3 |
| | 16 | 0.1 |
| | 17 | 0.6 |
| | 18 | 0.5 |
| | 19 | 0.4 |
| | 20 | 0.9 |

| ID | COSC32 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.1 |
| | 2 | 0.2 |
| | 3 | 0.1 |
| | 4 | 0.1 |
| | 5 | 0.8 |
| | 6 | 0.9 |
| | 7 | 0.5 |
| | 8 | 0.6 |
| | 9 | 0.9 |
| | 10 | 0.7 |
| | 11 | 0.9 |
| | 12 | 0.8 |
| | 13 | 0.5 |
| | 14 | 0.4 |
| | 15 | 0.6 |
| | 16 | 0.4 |
| | 17 | 0.8 |
| | 18 | 0.2 |
| | 19 | 0.9 |
| | 20 | 0.4 |

| ID | COSC33 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.3 |
| | 2 | 0.4 |
| | 3 | 0.7 |
| | 4 | 0.1 |
| | 5 | 0.8 |
| | 6 | 0.9 |
| | 7 | 0.4 |
| | 8 | 0.6 |
| | 9 | 0.3 |
| | 10 | 0.7 |
| | 11 | 0.1 |
| | 12 | 0.3 |
| | 13 | 0.9 |
| | 14 | 0.7 |
| | 15 | 0.8 |
| | 16 | 0.7 |
| | 17 | 0.5 |
| | 18 | 0.6 |
| | 19 | 0.1 |
| | 20 | 0.4 |

| ID | COSC34 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.2 |
| | 2 | 0.4 |
| | 3 | 0.5 |
| | 4 | 0.2 |
| | 5 | 0.8 |
| | 6 | 0.9 |
| | 7 | 0.7 |
| | 8 | 0.6 |
| | 9 | 0.3 |
| | 10 | 0.2 |
| | 11 | 0.5 |
| | 12 | 0.3 |
| | 13 | 0.1 |
| | 14 | 0.1 |
| | 15 | 0.2 |
| | 16 | 0.6 |
| | 17 | 0.5 |
| | 18 | 0.7 |
| | 19 | 0.5 |
| | 20 | 0.2 |

| ID | COSC35 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.2 |
| | 2 | 0.3 |
| | 3 | 0.4 |
| | 4 | 0.1 |
| | 5 | 0.5 |
| | 6 | 0.7 |
| | 7 | 0.4 |
| | 8 | 0.6 |
| | 9 | 0.3 |
| | 10 | 0.2 |
| | 11 | 0.5 |
| | 12 | 0.3 |
| | 13 | 0.9 |
| | 14 | 0.7 |
| | 15 | 0.9 |
| | 16 | 0.7 |
| | 17 | 0.3 |
| | 18 | 0.4 |
| | 19 | 0.1 |
| | 20 | 0.3 |

| ID | COSC36 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.5 |
| | 2 | 0.3 |
| | 3 | 0.4 |
| | 4 | 0.6 |
| | 5 | 0.9 |
| | 6 | 0.9 |
| | 7 | 0.7 |
| | 8 | 0.9 |
| | 9 | 0.3 |
| | 10 | 0.2 |
| | 11 | 0.5 |
| | 12 | 0.3 |
| | 13 | 0.8 |
| | 14 | 0.6 |
| | 15 | 0.9 |
| | 16 | 0.5 |
| | 17 | 0.4 |
| | 18 | 0.8 |
| | 19 | 0.2 |
| | 20 | 0.3 |

| ID | COSC37 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.8 |
| | 2 | 0.9 |
| | 3 | 0.2 |
| | 4 | 0.8 |
| | 5 | 0.8 |
| | 6 | 0.6 |
| | 7 | 0.7 |
| | 8 | 0.9 |
| | 9 | 0.5 |
| | 10 | 0.6 |
| | 11 | 0.5 |
| | 12 | 0.4 |
| | 13 | 0.2 |
| | 14 | 0.9 |
| | 15 | 0.2 |
| | 16 | 0.1 |
| | 17 | 0.3 |
| | 18 | 0.7 |
| | 19 | 0.2 |
| | 20 | 0.1 |

| ID | COSC38 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.5 |
| | 2 | 0.4 |
| | 3 | 0.6 |
| | 4 | 0.3 |
| | 5 | 0.8 |
| | 6 | 0.9 |
| | 7 | 0.5 |
| | 8 | 0.6 |
| | 9 | 0.9 |
| | 10 | 0.8 |
| | 11 | 0.7 |
| | 12 | 0.9 |
| | 13 | 0.4 |
| | 14 | 0.2 |
| | 15 | 0.1 |
| | 16 | 0.4 |
| | 17 | 0.6 |
| | 18 | 0.1 |
| | 19 | 0.3 |
| | 20 | 0.5 |

| ID | COSC39 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.2 |
| | 2 | 0.1 |
| | 3 | 0.5 |
| | 4 | 0.4 |
| | 5 | 0.7 |
| | 6 | 0.8 |
| | 7 | 0.5 |
| | 8 | 0.8 |
| | 9 | 0.5 |
| | 10 | 0.4 |
| | 11 | 0.9 |
| | 12 | 0.7 |
| | 13 | 0.7 |
| | 14 | 0.5 |
| | 15 | 0.9 |
| | 16 | 0.5 |
| | 17 | 0.4 |
| | 18 | 0.6 |
| | 19 | 0.4 |
| | 20 | 0.1 |

| ID | COSC40 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.6 |
| | 2 | 0.5 |
| | 3 | 0.7 |
| | 4 | 0.4 |
| | 5 | 0.5 |
| | 6 | 0.9 |
| | 7 | 0.8 |
| | 8 | 0.6 |
| | 9 | 0.6 |
| | 10 | 0.5 |
| | 11 | 0.4 |
| | 12 | 0.8 |
| | 13 | 0.6 |
| | 14 | 0.8 |
| | 15 | 0.4 |
| | 16 | 0.5 |
| | 17 | 0.8 |
| | 18 | 0.5 |
| | 19 | 0.5 |
| | 20 | 0.8 |

| ID | COSC41 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.8 |
| | 2 | 0.2 |
| | 3 | 0.5 |
| | 4 | 0.7 |
| | 5 | 0.8 |
| | 6 | 0.7 |
| | 7 | 0.3 |
| | 8 | 0.6 |
| | 9 | 0.5 |
| | 10 | 0.6 |
| | 11 | 0.8 |
| | 12 | 0.4 |
| | 13 | 0.1 |
| | 14 | 0.7 |
| | 15 | 0.9 |
| | 16 | 0.3 |
| | 17 | 0.9 |
| | 18 | 0.1 |
| | 19 | 0.2 |
| | 20 | 0.9 |

| ID | COSC42 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.2 |
| | 2 | 0.2 |
| | 3 | 0.7 |
| | 4 | 0.6 |
| | 5 | 0.8 |
| | 6 | 0.5 |
| | 7 | 0.9 |
| | 8 | 0.6 |
| | 9 | 0.3 |
| | 10 | 0.2 |
| | 11 | 0.5 |
| | 12 | 0.3 |
| | 13 | 0.4 |
| | 14 | 0.7 |
| | 15 | 0.1 |
| | 16 | 0.3 |
| | 17 | 0.4 |
| | 18 | 0.6 |
| | 19 | 0.7 |
| | 20 | 0.9 |

| ID | COSC43 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.9 |
| | 2 | 0.7 |
| | 3 | 0.9 |
| | 4 | 0.6 |
| | 5 | 0.1 |
| | 6 | 0.2 |
| | 7 | 0.2 |
| | 8 | 0.1 |
| | 9 | 0.4 |
| | 10 | 0.7 |
| | 11 | 0.3 |
| | 12 | 0.7 |
| | 13 | 0.8 |
| | 14 | 0.5 |
| | 15 | 0.6 |
| | 16 | 0.3 |
| | 17 | 0.4 |
| | 18 | 0.2 |
| | 19 | 0.5 |
| | 20 | 0.2 |

| ID | COSC44 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.4 |
| | 2 | 0.6 |
| | 3 | 0.3 |
| | 4 | 0.5 |
| | 5 | 0.9 |
| | 6 | 0.8 |
| | 7 | 0.6 |
| | 8 | 0.7 |
| | 9 | 0.3 |
| | 10 | 0.2 |
| | 11 | 0.5 |
| | 12 | 0.3 |
| | 13 | 0.5 |
| | 14 | 0.7 |
| | 15 | 0.5 |
| | 16 | 0.4 |
| | 17 | 0.2 |
| | 18 | 0.1 |
| | 19 | 0.1 |
| | 20 | 0.3 |

| ID | COSC45 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.1 |
| | 2 | 0.5 |
| | 3 | 0.2 |
| | 4 | 0.3 |
| | 5 | 0.8 |
| | 6 | 0.9 |
| | 7 | 0.7 |
| | 8 | 0.6 |
| | 9 | 0.5 |
| | 10 | 0.4 |
| | 11 | 0.6 |
| | 12 | 0.5 |
| | 13 | 0.9 |
| | 14 | 0.4 |
| | 15 | 0.9 |
| | 16 | 0.2 |
| | 17 | 0.6 |
| | 18 | 0.4 |
| | 19 | 0.7 |
| | 20 | 0.8 |

| ID | COSC46 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.8 |
| | 2 | 0.9 |
| | 3 | 0.8 |
| | 4 | 0.7 |
| | 5 | 0.6 |
| | 6 | 0.5 |
| | 7 | 0.7 |
| | 8 | 0.3 |
| | 9 | 0.2 |
| | 10 | 0.6 |
| | 11 | 0.5 |
| | 12 | 0.3 |
| | 13 | 0.2 |
| | 14 | 0.1 |
| | 15 | 0.3 |
| | 16 | 0.1 |
| | 17 | 0.2 |
| | 18 | 0.1 |
| | 19 | 0.4 |
| | 20 | 0.5 |

| ID | COSC47 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.5 |
| | 2 | 0.6 |
| | 3 | 0.4 |
| | 4 | 0.5 |
| | 5 | 0.8 |
| | 6 | 0.8 |
| | 7 | 0.7 |
| | 8 | 0.6 |
| | 9 | 0.9 |
| | 10 | 0.2 |
| | 11 | 0.8 |
| | 12 | 0.2 |
| | 13 | 0.1 |
| | 14 | 0.3 |
| | 15 | 0.1 |
| | 16 | 0.4 |
| | 17 | 0.6 |
| | 18 | 0.2 |
| | 19 | 0.4 |
| | 20 | 0.5 |

| ID | COSC48 | |
|---|---|---|
| The KW Table | ID | Level |
| | 1 | 0.8 |
| | 2 | 0.9 |
| | 3 | 0.7 |
| | 4 | 0.8 |
| | 5 | 0.5 |
| | 6 | 0.9 |
| | 7 | 0.7 |
| | 8 | 0.6 |
| | 9 | 0.5 |
| | 10 | 0.7 |
| | 11 | 0.4 |
| | 12 | 0.5 |
| | 13 | 0.1 |
| | 14 | 0.2 |
| | 15 | 0.1 |
| | 16 | 0.3 |
| | 17 | 0.3 |
| | 18 | 0.2 |
| | 19 | 0.1 |
| | 20 | 0.3 |

| **ID** | COSC49 | |
|---|---|---|
| **The KW Table** | **ID** | **Level** |
| | 1 | 0.8 |
| | 2 | 0.1 |
| | 3 | 0.9 |
| | 4 | 0.8 |
| | 5 | 0.1 |
| | 6 | 0.2 |
| | 7 | 0.1 |
| | 8 | 0.4 |
| | 9 | 0.6 |
| | 10 | 0.7 |
| | 11 | 0.4 |
| | 12 | 0.5 |
| | 13 | 0.2 |
| | 14 | 0.3 |
| | 15 | 0.3 |
| | 16 | 0.4 |
| | 17 | 0.6 |
| | 18 | 0.3 |
| | 19 | 0.4 |
| | 20 | 0.5 |

| **ID** | COSC50 | |
|---|---|---|
| **The KW Table** | **ID** | **Level** |
| | 1 | 0.1 |
| | 2 | 0.9 |
| | 3 | 0.3 |
| | 4 | 0.4 |
| | 5 | 0.8 |
| | 6 | 0.9 |
| | 7 | 0.7 |
| | 8 | 0.6 |
| | 9 | 0.6 |
| | 10 | 0.5 |
| | 11 | 0.7 |
| | 12 | 0.3 |
| | 13 | 0.2 |
| | 14 | 0.1 |
| | 15 | 0.2 |
| | 16 | 0.6 |
| | 17 | 0.4 |
| | 18 | 0.3 |
| | 19 | 0.5 |
| | 20 | 0.7 |

# BIBLIOGRAPHY


[1]. Mark White, *"SmartBadge: An Electronic Conference Badge using RF and IR Communications"*, the thesis of Engineering Master degree, University of Canterbury.

[2]. Richard Borovoy, Fred Martin, Sd Vemuri, Mitchel Resnick, Brian Sflverman, and Chris Hancock, *"Meme Tags and Community Mirrors: Moving from Conferences to Collaboration",* CSCW98, Seattle Washington USA, 1998, 1:159-168.

[3]. Reynolds, C. W. (1999), *"Steering Behaviors For Autonomous Characters"*, in the proceedings of Game Developers Conference 1999 held in San Jose, California. Miller Freeman Game Group, San Francisco, California. Pages 763-782.

[4]. Carol O'Sullivan, Justine Cassell, Hannes Vilhjálmsson, Simon Dobbyn, Christopher Peters, William Leeson, Thanh Giang and John Dingliana, *"Crowd and Group Simulation with Levels of Detail for Geometry, Motion and Behaviour",* Trinity College Dublin, MIT Media Lab, third Irish Workshop on Computer Graphics (2002), Eurographics Irish Chapter.

[5]. Networking Application, http://www.ntag.com/products/applications/commonground.html.

[6]. Flavia Sparacino, *"The Museum Wearable: real-time sensor-driven understanding of visitors' interests for the personalized visually-augmented museum experience",* accepted for publication in Proceedings of: Museums and the Web (MW2002), Boston, April 17-20, 2002.

[7]. Branislav Ulicny and Daniel Thalmann, *"Crowd simulation for interactive virtual environments and VR training systems",* Computer Graphics Lab (LIG), Swiss Federal Institute of Technology, Switzerland. Proc. Eurographics Workshop on Animation and Simulation, 2001.

[8]. Franck FEURTEY, *"Simulating the Collision Avoidance Behaviour of Pedestrians"*, Department of Electronic Engineering, 15/02/2000

[9]. Finn V. Jensen, "*Bayesian Networks and Decision Graphs*", 2001 Springer-Verlag New York, Inc.

[10]. Schmid Martin, Kneubuehl Thomas, *"CSF Crowd Simulation Framework - Document"*, University of Applied Sciences, Biel/Bienne, Switzerland, Computer Graphics Diploma Project.

[11]. Stuart Russell, Peter Norvig. "*Artificial Intelligence – A Modern Approach*", First Edition.

[12]. Craig W. Reynolds*, "Flocks, Herds, and Schools: A Distributed Behavioural Model",* Computer Graphics 21(4), July, 1987, edited by Maureen C. Stone, pages 25-34.

[13]. http://www.red3d.com/cwr/steer/PathFollow.html

[14]. http://www.red3d.com/cwr/steer/Doorway.html

[15]. http://www.red3d.com/cwr/steer/SeekFlee.html

[16]. http://www.red3d.com/cwr/steer/Wall.html

[17]. http://www.red3d.com/cwr/steer/Wander.html

[18]. http://www.red3d.com/cwr/steer/Obstacle.html

[19]. Aris M. Ouksel, Yair M. Babad and Thomas Tesch, *"Matchmaking Software Agents in B2B Markets"***,** the 37th Hawaii International Conference on System Sciences, 2004.

[20]. AnHai Doan,Ying Lu,Yoonkyong Lee, and Jiawei Han, "*Profile-Based Object Matching for Information Integration*", IEEE Computer Society, 2003.

[21]. Alex Borgida, Peter F. Patel-Schneider, "*A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic*", Journal of Artificial Intelligence Research 1, pp 277 – 388, 1994.

[22]. Maja Milicic, *"Description Logics with Concrete Domains and Functional Dependencies"*, Master's Thesis, Technical University 32 Dresden, Department of Computer Science, 2004.

[23]. F. Baader and W. Nutt. "*Basic Description Logics"*. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors, The Description Logic Handbook: Theory, Implementation, and Applications, pages 43--95. Cambridge University Press, 2003.

[24]. http://www.cosc.canterbury.ac.nz/teaching/handouts/cosc401/L2_fundamentals.pdf

[25]. http://www-cs-students.stanford.edu/~pdoyle/quail/notes/pdoyle/learning.html

[26]. David Heckerman, "*A Tutorial on Learning with Bayesian Networks*", March 1995, Technical Report, MSR-TR-95-06 Microsoft Research Advanced Technology Division Microsoft Corporation,

[27]. ftp://ftp.research.microsoft.com/pub/dtg/david/tutorial.ps

[28]. http://www.cs.huji.ac.il/~nir/Nips01-Tutorial/

[29]. http://en.wikipedia.org/wiki/Bayesian_inference

[30]. http://www.cis.strath.ac.uk/research/seminars/0210_Fernandez_Luna.html

[31]. B. Sierra and P. Larra.naga, "*Predicting the Survival in Malignant Skin Melanoma using Bayesian Networks. An Empirical Comparison Between Different Approaches*", Artificial Intelligence in Medicine, 14(1-2), 215. 230, (1998).

[32]. S. L. Lauritzen and D. J. Spiegelhalter. "*Local Computations with Probabilities on Graphical Structures and Their Applications to Expert Systems*", Proceedings of the Royal Statistical Society, Series B., 50, 154-227. 1988.

[33]. Pearl, J. Fusion, "*Propagation and Structuring in Belief Network"*. UCLA Computer Science Department Technical Report 850022 (R-42); Artificial Intelligence, Vol. 29, No.3, 241-288, September 1986.

[34]. R. Shachter. "*Intelligent Probabilistic Inference*". In J. F. Lemmer and L. N. Kanal, editors, Uncertainty in Artificial Intelligence, pages 371 – 382. North Holland, Amsterdam, 1986.

[35]. http://opensteer.sourceforge.net/doc.html

[36]. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/winsock/windows_sockets_start_page_2.asp

[37]. Andrea Calì, Diego Calvanese, Simona Colucci,Di Noia, Tommaso and Francesco M. Donini, "*A Description Logic Based Approach for Matching User Profiles*", Proc. of the 2004 Description Logic Workshop (DL 2004).

[38]. Flavia Sparacino, *"Sto(ry)chastics: a Bayesian Network Architecture for User Modeling and Computational Storytelling for Interactive Spaces"*, Proceedings of Ubicomp, The Fifth International Conference on Ubiquitous Computing 2003: Seattle, WA, USA.